INPUT

Publicación práctica para usuarios de

sinclair

Revista mensual 1987 Precio 375 Ptas Año 2 Número 19

MAPA Y CARGADOR DEL FUTURE KNIGHT

TODO SOBRE LAS RUTINAS DE GRABACION

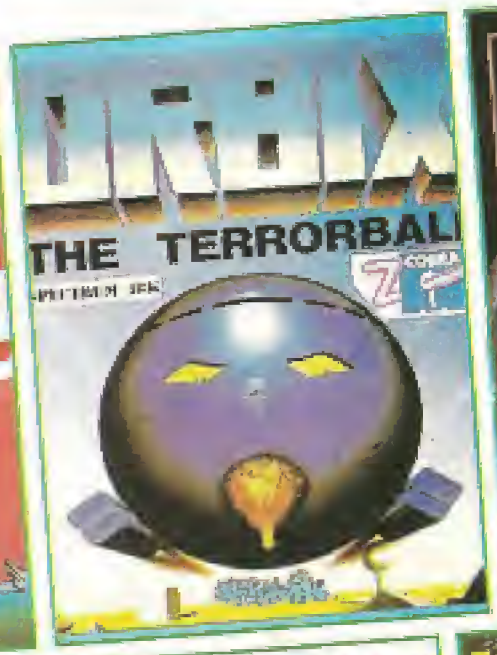
EL JUEGO DEL ZORRO Y LAS OCAS

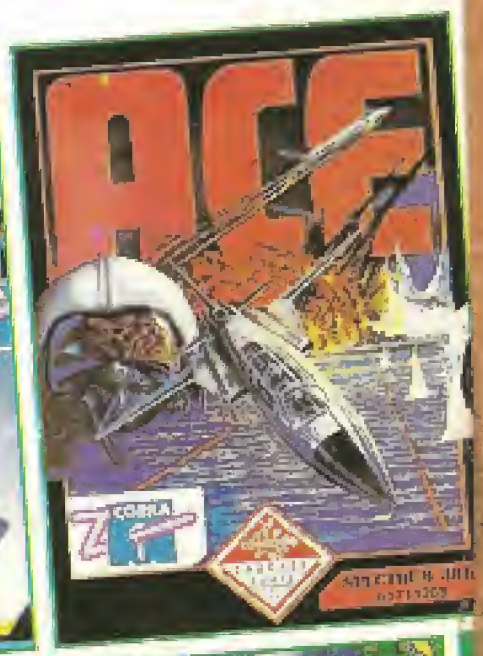
CALCULO DE POTENCIAS

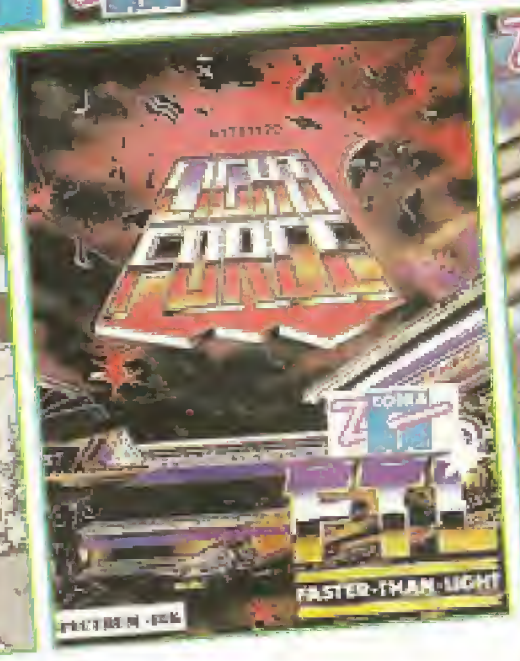
Nueva sección para el 128

CINCO MINUTOS ANTES DE COMPRAR UN JUEGO A 875 P

ECHALE UN VISTAZO A ESTOS JUEGOS DE 875 Ptas.







875 Ptas.

VERSION CASSETTE

COBRA SOFTWARE

SÍGUENOS EL JUEG

ZAFIRO SOFTWARE DIVISION Paseo de la Castellana, 141 28046 Madrid
Tel. 455 30 04 Tel. Barna. 209 33 55 Telex: 22690 ZAFIR E

AÑO 2 NUMERO 19

DIRECTOR: Manuel Pérez
DIRECTOR DE ARTE: Luis F. Balaguer
REALIZACION GRAFICA: Didac Tudela
COLABORADORES: Jose Vila, Antonio Priego, Xavier Ferrer, Ernesto del Valle, Equipo Mafisoft, Ramón Rabaso, Antonio Tarabiel, Jaime Mardones, Carlos Bartolomé, Angels Alvarez
FOTOGRAFIA: Ernesto Walfisch, Joan Boada

INPUT Sinclair es una publicación de PLANETA-DE AGOSTINI, S.A.
GERENTE DIVISION DE REVISTAS: Sebastián Martínez

PUBLICIDAD: José Real-Grupo Jota
Madrid: c/ General Varela, 35
Telef. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.°
Telef. 250 23 99

FOTOMECANICA: TECFA S.A.

IMPRESION: Sirven Grafic
c/ Gran Vía, 754-756, 08013 Barcelona
Depósito legal: B. 38.115-1986

SUSCRIPCIONES: EDISA
López de Hoyos, 141 28002 Madrid
Telef. (91) 415 97 12

REDACCION:
Aribau, 185, 1.°
08021 Barcelona

DISTRIBUIDORA:
B.B.A. PROMOTORA DE EDICIONES, S.A.
Calle G, n.° 11, Sector B, Zona Franca
08004 Barcelona

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea

INPUT Sinclair es una publicación controlada por

INPUT Sinclair es independiente y no está vinculada a Sinclair Research o sus distribuidores

INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío. Las respuestas se canalizarán a través de las secciones adecuadas en estas páginas

INPUT sinclair

SUMARIO

EDITORIAL

NUEVAS NECESIDADES DE LOS USUARIOS

A medida que pasa el tiempo y los usuarios conocen mejor los equipos con los que trabajan, sus demandas hacia los medios de comunicación especializados, también se modifican.

INPUT no podía permanecer insensible a esas nuevas tendencias, tan naturales como el paso del tiempo, y decidió ir renovándose a sí misma.

Ya habréis observado que tanto en algunos contenidos como en las presentaciones, en números anteriores, se han introducido novedades. En este ejemplar que ahora tenéis en vuestras manos los cambios ya afectan a una parte considerable de la revista.

De entrada debemos aclarar que en ningún caso se pretende romper con una tradición que ha conectado muy bien con los deseos de nuestros lectores, como nos consta por la aceptación y venta de INPUT y por las numerosas cartas que en la redacción se reciben todos los meses.

Sí se anhela, en cambio, dar cabida a esas nuevas necesidades que antes mencionábamos. No es un reto fácil, pues se trata de intentar conciliar cosas aparentemente tan contradictorias como la información más completa sobre videojuegos y todo el entorno que ello comporta, por un lado, y el diseño y presentación de utilidades, aplicaciones y nociones de programación de forma educativa, clara, comprensible y amena, por otro.

Esta necesidad permanente de formación se inscribe en el marco de una oferta informática que siempre ofrece más a un menor coste. Lógicamente, quienes hoy comienzan con pequeños micros domésticos operarán mañana con potentes ordenadores personales. En esta perspectiva el poso de conocimientos generado en la primera fase será de incalculable utilidad en el futuro.

Así pues, el mundo de la informática, y aunque con rasgos propios la microinformática doméstica forma parte de él, tiene que construir una nueva forma de hacer que forzosamente pasa por conseguir que su presencia sea imprescindible tanto a la hora de trabajar como a la de jugar. En este empeño INPUT continúa a vuestro servicio pensando siempre en ofreceros cosas nuevas a la vez que prácticas.

Vosotros sois quienes finalmente decidiréis si, con tales fundamentos, INPUT responde plenamente a vuestras demandas.

NUEVA
REVISTA MENSUAL

isaac ASIMOV

selecciona para ti los mejores relatos de

CIENCIA FICCION

A-062

EL AGITADO MUNDO DEL 128

En este número iniciamos una nueva sección dedicada al SPECTRUM PLUS 2, sus periféricos, y especialmente al software 128 K.

LA CAIDA DEL IMPERIO SINCLAIR

Cuando **Alan Sugar** se hizo con las riendas del problemático imperio de **Sir Clive Sinclair**, su primera medida fue promover el diseño de un nuevo micro doméstico de bajo precio, que dispusiera de mayor capacidad de memoria y mejores prestaciones que los modelos precedentes, a semejanza de los ordenadores AMSTRAD, con los que tantos éxitos había cosechado hasta entonces. Para garantizar una buena acogida entre el público y asegurar su rentabilidad, se señaló que este nuevo modelo debía ser compatible con todo el soft Sinclair de 48 K existente en el mercado, además de incorporar la posibilidad de desarrollar su propio soft específico para 128 K.

Así nació el **SPECTRUM PLUS 2**, con su teclado profesional, su cassette incorporado, y sus flamantes 16 chips de RAM.

EL PROBLEMA DEL SOFTWARE DE 128 K

El principal problema con que se han enfrentado hasta ahora los usuarios del **SPECTRUM PLUS 2**, es la imposibilidad de sacar a sus 128 K todo el partido posible, debido fundamentalmente al escaso volumen de soft específico publicado hasta el momento. Esta circunstancia se traduce en una curiosa situación que pudo poner en peligro el futuro del **PLUS 2** durante las pasadas navidades, cuando las expectativas de **AMSTRAD-SINCLAIR** se vieron frustradas por un mercado reticente.

Por un lado, las compañías de software no querían arriesgarse a producir y promocionar programas de 128 Kylobites a gran escala, en tanto no aumentaran las ventas del **SPECTRUM PLUS 2**; y por otro, los usuarios no estaban dispuestos a invertir sus ahorros en un nuevo micro mientras no hubiera soft específico disponible.

Anticipándose a esta situación, el sagaz Alan Sugar consiguió arrancar a varias firmas productoras de software —OCEAN entre ellas— la promesa de poner a la venta un número importante de programas de 128 K a corto plazo. Aunque este compromiso aún no ha sido cumplido plenamente, todo parece indicar que muy pronto dejará de haber motivos que justifiquen las reservas de los usuarios. Buena prueba de ello es la aparición en el Reino Unido de varios títulos interesantes, algunos de los cuales pronto serán publicados en España, que rápidamente van creciendo en número e incorporándose a las listas de éxitos. A continuación os hablaremos de algunos de ellos con más detalle.

PROGRAMAS DISPONIBLES

Hasta ahora, la fórmula preferida por los fabricantes para satisfacer las exigencias de los usuarios del **SPECTRUM PLUS II**, consiste en diseñar un programa de 48 K, promocionarlo, y después presentar una versión *ampliada* a 128 K, con más pantallas y mejores efectos de sonido.

Sin duda, el ejemplo más original lo tenemos en el sistema puesto en práctica por la joven compañía **OPERA SOFT**, que consiste en incluir en cada cinta una versión de 48 K grabada en la cara "A", y otra mejorada de 128 K en la cara "B".

Otro caso completamente diferente es el de **THE EDGE**, en su sensacional **FAIRLIGHT II**. Este programa puede cargarse por partes en un 48 K, añadiendo nuevas fases a medida que se va completando el juego, o de una vez, en un **SPECTRUM PLUS II** en modo 128 K. Sin embargo, el contenido es exactamente el mismo en uno y otro caso.

Por el momento, son muy escasos los títulos disponibles en versión única de 128 K, y de ellos ninguno está a la venta en España, aunque algunos usuarios se las han sabido arreglar para conseguirlos. Entre los que mayor impacto han causado en Londres, podríamos mencionar **THEY TELL ME TROOPER** —una original aventura espacial—, **GALLIPOLI** (el mejor *wargame* para micro diseñado hasta la fecha), y un magnífico arcade de OCEAN.

Existen también numerosas versiones 128 K de programas inicialmente diseñados para el SPECTRUM 48 K, como **THEY STOLE A MILLION**, disponible sólo en el Reino Unido, y dos programas **MASTERTRONIC** que comentamos en nuestra revista de software y que muy pronto serán publicados en nuestro país por la compañía **DRO SOFT: KNIGHT TIME** y **SPELL BOUND.**

¿Y AHORA EL PLUS 3?

Pues sí. Cuando aún no han sido digeridas ni aprovechadas todas las posibilidades del PLUS 2, el dinámico grupo AMSTRAD-SINCLAIR está preparando un nuevo producto que promete remover los cimientos de ese extenso micromundo formado por la familia Spectrum.

Según parece, el próximo verano será presentado en Gran Bretaña el nuevo Spectrum 128 K PLUS 3. La gran novedad que aportará este aún no nato ordenador será la de incorporar, en el lugar en el que el PLUS 2 lleva el cassette, una flamante unidad de disco.

Su aspecto exterior tenderá a asemejarse al del modelo 6128 de AMSTRAD, que también posee una unidad de disco en su lado derecho, mientras que el teclado será igual al del actual 128 PLUS 2.

Los discos serán de 3 pulgadas y correrán con la nueva versión del sistema operativo Ams-Dos, con ligeras modificaciones, efectuadas por Locomotive, para su adecuado funcionamiento en el Spectrum.

Se supone que su compatibilidad con el software existente para el 48 y el PLUS 2 será total. Para ello el futuro PLUS 3 irá equipado con las correspondientes entradas para Ear y Mic, a fin de que puedan cargarse todas las cintas. Asimismo ,podrán usarse los tradicionales Microdrives, con los que tantas cosas han hecho muchos de los perseverantes usuarios de Sinclair. El ordenador, por sí mismo, detectará cuándo recibe información de uno u otro tipo de soporte.

No se descarta tampoco que, en un futuro, fabricantes ajenos a AMSTRAD ensayen el desarrollo de la compatibilidad del PLUS 3 con CPM.

¿Y EL SOFTWARE?

Al comenzar estas líneas hacíamos referencia a los problemas que han encontrado los usuarios del 128 PLUS 2 a la hora de gozar a fondo de las posibilidades que en teoría les ofrecía su ordenador. ¿Ocurrirá lo mismo cuando se comercialice el Spectrum PLUS 3?

Las posibilidades que este nuevo ordenador ofrece nos hacen suponer que la historia no se repetirá. Aun cuando en un primer momento la edición de juegos y utilidades para el PLUS 3 no pasará de ser un revival adaptado del software que ya corre para 48, la utilización de discos abre un enorme campo de posibilidades.

No se trata única y exclusivamente de las grandes ventajas que supondrá poder hacer programas más largos, complejos, en los que será posible utilizar toda clase de recursos decorativos y efectistas. La rapidez e interactividad que implica la utilización del disco permitirá utilizar en cada momento únicamente lo que nos interese, por lo que el rendimiento de la capacidad de memoria será mucho mayor. A estas alturas algunos ya estarán gozando ante su pantalla y recreándose con juegos del más alto nivel de cualquier sistema; como por ejemplo, los de Commodore.

Tampoco debemos olvidar que no sólo en los juegos notaremos la diferencia. Programas de utilidades como los procesadores de textos, las bases de datos o las contabilidades, cuyo uso con el cassette resultaba poco menos que engorroso, correrán en el PLUS 3 con fluidez pasmosa.

UNA POSITIVA RESPUESTA

De la misma opinión parecen ser los principales productores de software, que ya han respondido positivamente a las llamadas de AMSTRAD y han ofrecido su apoyo logístico a fin de que el lanzamiento del PLUS 3 se vea arropado por un adecuado paquete de programas. Entre estas casas destaca OCEAN, quien ya adquirió compromisos de producción con AMSTRAD en relación con el PLUS 2. Sin duda se avecinan grandes cambios para los fieles amantes del Spectrum.

En Gran Bretaña se especula con un precio de lanzamiento realmente interesante: alrededor de las 40.000 pesetas.

Según ha podido averiguar INPUT, el lanzamiento del PLUS 3 en nuestro país se producirá hacia finales de este año.

LA ELEVACION A POTENCIAS

Aquí tienes unas cuantas funciones matemáticas que tienen una gran cantidad de aplicaciones prácticas, desde la medida de la superficie de tu casa hasta el cálculo de la velocidad de un cuerpo que cae.

La **potenciación** o **función potencial** es una de las funciones matemáticas a las que con frecuencia no se les presta mucha atención y que, sin embargo, tiene una sorprendente variedad de usos, especialmente en los casos en que quieras calcular áreas o volúmenes en tus programas.

La función potencial se representa con el signo ↑ en el ordenador. Se coloca entre dos números, 2↑3, por ejemplo, y se lee «dos elevado a la tercera potencia», o «dos elevado al cubo». (El código ASCII de la flecha ↑ es **alt-24**.)

Aunque al principio te pueda parecer confuso, en realidad la idea es muy simple. Un número elevado a una determinada potencia, que es un segundo número, es simplemente el primer número multiplicado por sí mismo tantas veces como indique el segundo número.

Según esto, si volvemos al ejemplo anterior, dos elevado a la tercera potencia es lo mismo que dos multiplicado por dos tres veces; así:

```
2↑3=2*2*2=8
```

Es igual para cualquier par de números.

Por ejemplo:

```
2↑5=2*2*2*2*2=32
5↑4=5*5*5*5=625
```

También puedes escribir los números en la forma 2^2, 2^3, 2^5 y 5^4, que es la notación convencional para las matemáticas. Pero esta forma no la entiende el ordenador.

La segunda y tercera potencia tienen nombre especial. Cualquier número elevado a la segunda potencia se dice que está elevado «al cuadrado».

Análogamente, cualquier número elevado a la tercera potencia se dice que está elevado «al cubo».

Realmente existen razones sólidas para utilizar los nombres asociados con estas dos potencias.

Si intentas medir el área de un rectángulo (ya se trate de un trozo de césped, una alfombra o cualquier otra entidad que tenga forma rectangular), lo que haces es medir su longitud y su anchura y, a continuación, multiplicar una medida por otra.

El resultado de este cálculo es un número de unidades «cuadradas». La razón de que un área se mida, por ejemplo, en metros cuadrados reside en que lo que realmente estás haciendo es medir cuántos cuadrados de un metro de lado caben en la superficie que estás midiendo.

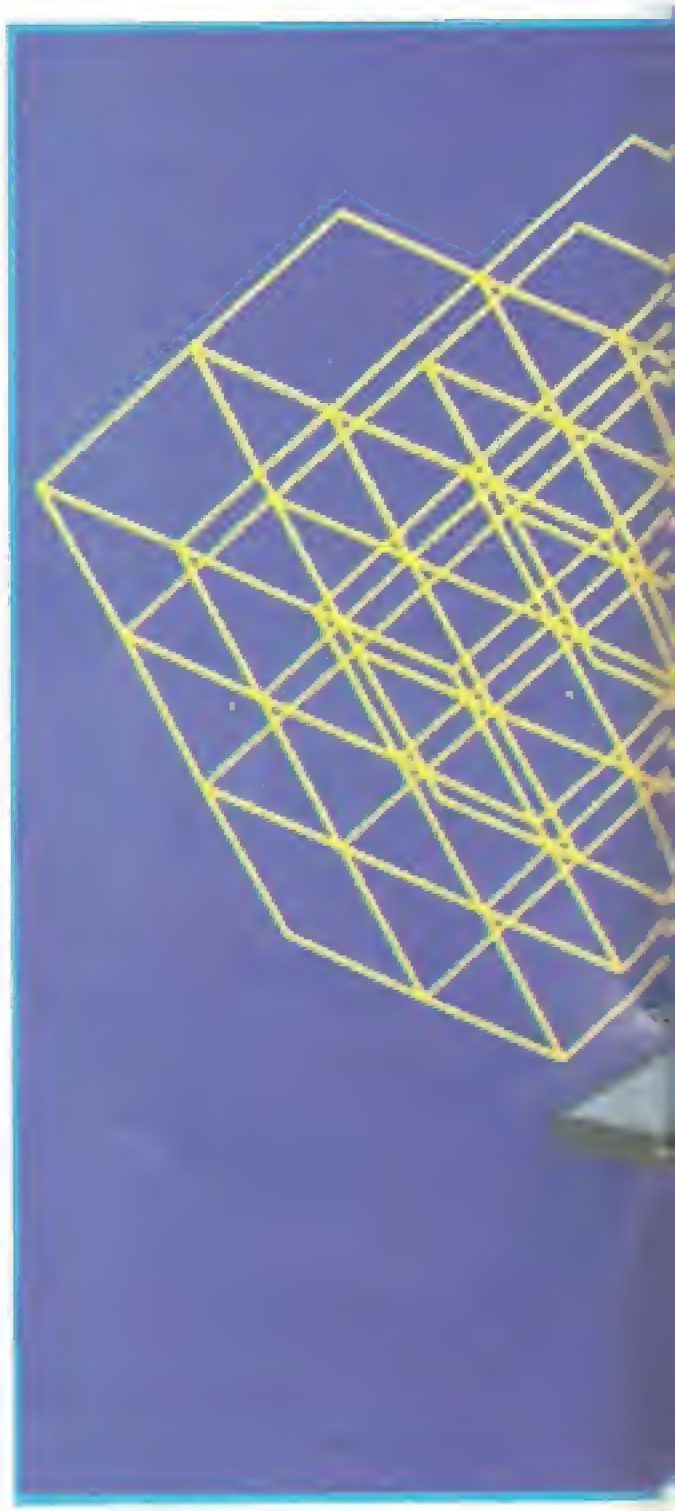
Análogamente, un número elevado a la segunda potencia se dice que está elevado al cuadrado; la unidad de superficie o unidad cuadrada se representa por la multiplicación de la unidad básica por sí misma.

De la misma forma que la segunda potencia toma su nombre de la medida de superficies, la tercera potencia toma el suyo de la medida de volúmenes. Para encontrar el volumen de un cubo, se multiplica su anchura por su longitud y por su altura. Por eso hacen falta tres multiplicaciones de la unidad básica para encontrar el volumen (dos para la superficie y una más para la altura); por ello se dice que un número elevado a la potencia tres está elevado «al cubo».

MAS POTENCIAS

A partir de aquí, ya no es posible construir un modelo acerca de lo que significa la elevación a potencias. Si fuera posible tener un objeto que tuviera cuatro dimensiones, su volumen se mediría en unidades fundamentales elevadas a la cuarta potencia, lo cual es obviamente absurdo. Sin embargo, hay muchos cálculos en los que intervienen estas potencias superiores y resultan útiles por otras razones.

Por ejemplo, supongamos que quieres conocer la probabilidad de que al tirar un dado al aire salga como resul-

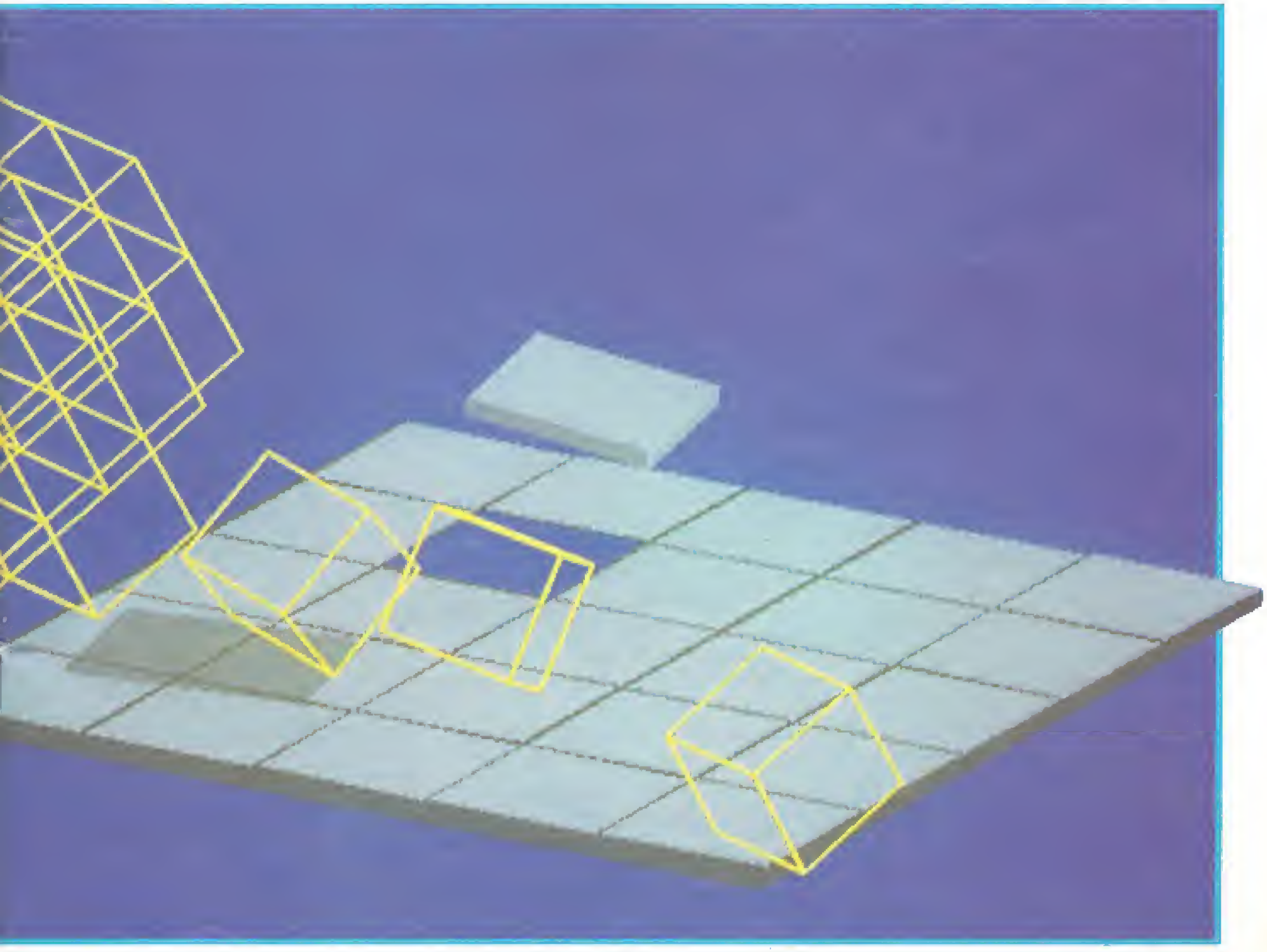


- LA FUNCION POTENCIAL
- EL CUADRADO Y EL CUBO DE UN NUMERO
- ELEVACION DE NUMEROS A POTENCIAS SUPERIORES
- HALLANDO EL AREA DE UN CUADRADO
- RAICES CUADRADAS
- USO DE LA FUNCION SQR EN UN PROGRAMA

tado un seis. Es fácil: si suponemos que todas las caras tienen la misma probabilidad de quedar hacia arriba, la probabilidad de obtener un determinado número en particular es $\frac{1}{6}$.

Pero supongamos ahora que quieres conocer la probabilidad de obtener dos veces seguidas un 6. Tienes un sexto de probabilidades de que la primera vez te salga un 6, y si esto sucede tienes otra vez un sexto de probabilidades de que la siguiente vez salga un 6. Por ello la probabilidad total es $\frac{1}{6}$ veces $\frac{1}{6}$. En otras palabras, es $\frac{1}{6}$ elevado al cuadrado. Este mismo principio se aplica para cualquier número de tiradas; así la probabilidad de que al tirar un dado siete veces seguidas aparezca las siete veces un 7 es $\frac{1}{6}*\frac{1}{6}*\frac{1}{6}*\frac{1}{6}*\frac{1}{6}*\frac{1}{6}*\frac{1}{6} = (\frac{1}{6})\uparrow 7$, que, por cierto, es un número bastante pequeño.

Seguro que por lo menos has visto antes un caso de números elevados a potencias altas; te habrá aparecido en la conversión de números a binario. Cada dígito binario representa una potencia del número 2. Así en el número 11111111 el primer dígito de la derecha (el bit 0) representa un 1, el segundo (bit 1) representa un 2, el tercero (bit 2) representa un 4, el cuarto



(bit 3) representa un 8, y así sucesivamente 16, 32, 64 y 128.

Fíjate un poco más detalladamente: 4 es 2*2, o, lo que es lo mismo, 2 elevado a la potencia 2; 8 es 2*2*2 o 2 elevado a la potencia 3; 16 es 2*2*2*2, etcétera:

```
2 ↑ 2=2*2          =4
2 ↑ 3=2*2*2        =8
2 ↑ 4=2*2*2*2      =16
2 ↑ 5=2*2*2*2*2    =32
2 ↑ 6=2*2*2*2*2*2  =64
2 ↑ 7=2*2*2*2*2*2*2 =128
```

En la tabla anterior faltan dos valores. Aunque es muy fácil descubrir de qué valores se trata, la explicación puede resultar un poco sorprendente.

El primero de éstos es 2 ↑ 1. Es evidente que, de acuerdo a la analogía binaria, debe ser un 2. De hecho, por extensión, si 2 ↑ 2 es 2 multiplicado por sí mismo una vez, 2 ↑ 1 debería ser 2 multiplicado por sí mismo una vez menos. Es decir, 2 *no* se multiplica por sí mismo en absoluto, permaneciendo como 2.

El valor siguiente tal vez sea aún más sorprendente: 2 ↑ 0. Podría ser que te esperaras que tuviera el valor 0, pero si examinas los valores de la tabla anterior, observarás que le corresponde claramente el valor 1. Esto se debe a que ha de ser 2 multiplicado por sí mismo una vez. 1 es ya el número 2 una sola vez, la única manera en que se puede multiplicar por 2 una vez menos es dividiendo por 2. Y cualquier número dividido por sí mismo es 1.

Para comprobar todo esto u otros valores cualesquiera de potencias que desees experimentar, teclea lo siguiente:

```
PRINT 2 ↑ X
```

donde X es el valor que quieres comprobar; pulsa a continuación ENTER RETURN. Puedes probar con el valor que quieras (aunque con valores muy grandes puede que se te produzcan errores de desbordamiento). También puede ser que llegues a resultados bastante interesantes, por ejemplo, ¿qué sucedería si teclearas un valor fraccionario para la X, tal como el 0.5? Volveremos más adelante sobre este problema, pero primero nos detendremos un poco más en las segundas potencias o cuadrados, que se utilizan ampliamente.

DIBUJANDO CUADRADOS

El siguiente programa toma una serie de números que emplea para dibujar en pantalla una serie de cuadrados, y muestra la longitud de cada cuadrado y su área. A continuación te permite tomar dos cuadrados cualesquiera y comparar sus áreas. También te servirá para comprobar lo rápidamente que crecen los cuadrados a medida que lo va haciendo el número base. Por último te permitirá formarte una idea más global y significativa de las potencias.

```
1Ø CLS
2Ø PRINT AT 2,Ø;"Longitud";AT
   3,Ø;"del";AT 4,Ø;"lado"
3Ø PRINT AT 2,24;"Area";AT 3,
   24;"del";AT 4,24;"cuadrado"
4Ø LET z=Ø: PLOT 55,23
5Ø FOR n=1 TO 13
6Ø LET m=n: IF m>7 THEN LET
   m=m-8
7Ø DRAW Ø,n*8
8Ø DRAW PAPER m;n*8,Ø
9Ø DRAW PAPER m;Ø,-(n*8)
1ØØ DRAW PAPER 7;-(n*8),Ø
11Ø PRINT AT 6+(13-n),Ø;n;AT
   6+(13-n),25;n*n
12Ø PAUSE 1Ø: NEXT n
14Ø INPUT "te gustaria comparar
   cualquier area? (s/n)";a$
15Ø IF a$<>"s" AND a$<>"n"
   THEN GO TO 14Ø
16Ø IF a$="n" THEN GO TO 3ØØ
17Ø INPUT "cual es el primer
   cuadrado cuya area quieres
   comparar? (1-13)";a: IF
   a<1 OR a>13 THEN GO TO
   17Ø
18Ø INPUT "y cual es la
   segunda? (1-13)";b: IF
   b<1 OR b>13 THEN GO TO
   18Ø
19Ø LET x=INT (a): LET y=INT (b)
2ØØ IF x>y THEN GO TO 28Ø
```

```
21Ø CLS : PRINT "La primera
     area se ajusta en la segunda
     ";y^2/x^2;" veces."
22Ø PLOT 2Ø,Ø: DRAW x*8,Ø:
     DRAW Ø,x*8: DRAW -x*8,
     Ø: DRAW Ø,-x*8: DRAW
     y*8,Ø: DRAW Ø,y*8: DRAW
     -y*8,Ø: DRAW Ø,-y*8
23Ø PRINT "quieres comparar
     alguna mas? (s/n)"
24Ø INPUT a$
25Ø INPUT a$<>"s" AND
     a$<>"n" THEN GO TO 24Ø
26Ø IF a$="n" THEN LET z=1:
     GO TO 3ØØ
27Ø GO TO 17Ø
28Ø CLS : PRINT "La primera
     area es ";x^2/y^2;" veces
     mas grande que la segunda."
29Ø GO TO 22Ø
3ØØ IF z=1 THEN CLS : LET
     z=Ø
31Ø PRINT INK 2; FLASH 1;AT
     Ø,Ø;"Pulsa una tecla para
     verlo de nuevo"
32Ø PAUSE Ø
33Ø IF INKEY$="n" THEN STOP
34Ø RUN
```

El ordenador va dibujando un cuadrado cada vez, empezando por el más pequeño. La rutina que comienza en la línea 140 te permite comparar las áreas de dos cuadrados. Espera a que introduzcas un número, correspondiente al del primer cuadrado que deseas comparar, efectuando una comprobación para asegurarse de que efectivamente se trata de un número «legal». En otras palabras, si el número es más grande o más pequeño que el número de cuadrados, no será aceptado. Si introduces una fracción decimal, el ordenador la convertirá en un número entero por medio de la función INT.

Una vez que hayas introducido dos números válidos, el ordenador examina cuál de ellos es más grande. A continuación eleva cada número al cuadrado (multiplicándolo por sí mismo) y divide el mayor por el menor. Seguidamente se presenta en pantalla el mensaje que indica qué cuadrado es el más grande y cuántas veces superior. Nuevamente se te ofrecerá la posibilidad de comparar dos áreas, si deseas hacerlo o, por el contrario, de ver otra vez el programa completo.

Probablemente la potencia que más se utiliza es el cuadrado o segunda potencia. Pero también es posible que haya muchas veces en que quieras calcular el cuadrado al revés, es decir, encontrar el número original a partir del cual resulta un cuadrado conocido. Por ejemplo, si sabes que el área de un cuadrado es 81, puede ser que quieras calcular cuánto mide de largo cada lado.

En el caso del número 81 no resulta demasiado difícil: sabes que nueve veces nueve son 81, por lo que la longitud del lado buscada debe ser 9. Pero si el área del cuadrado buscado hubiera sido un número menos evidente, por ejemplo 127, resultaría bastante más laborioso calcular la medida del lado. Para ayudarte en esta tarea, los ordenadores disponen de una función especial: la raíz cuadrada.

Teclea en tu ordenador PRINT SQR(81) y a continuación pulsa EN-

TER o RETURN. Observarás que en la pantalla aparece el número 9. (En el caso de que sientas curiosidad por saber cuál es la raíz cuadrada de 127, sustituye en la expresión anterior 81 por 127.)

El ordenador dispone del comando especial SQR debido a que la raíz cuadrada se utiliza con mucha frecuencia, pero ésta no es de hecho la única forma en que se puede calcular la raíz cuadrada; también puede emplear la función potencial ordinaria. Si has estado experimentando en esta función con 2 ↑ 5, es decir, 2 elevado a $\frac{1}{2}$, la potencia $\frac{1}{2}$ tiene el mismo resultado que SQR(2). Para comprobar esto, ensaya PRINT SQR(81), después PRINT 81 ↑ .5.

Puede que los resultados no sean exactamente iguales, pero es seguro que serán muy parecidos.

Puedes realizar una operación análoga con la fracción $\frac{1}{3}$, que te permite obtener la función inversa de la elevación al cubo; esto es lo que se llama «extraer la raíz cúbica». Naturalmente puedes hacer lo mismo con otras potencias cualesquiera. Aunque muchas de ellas tienen sus aplicaciones, la más ampliamente utilizada de todas es, con mucha diferencia, la raíz cuadrada.

Puedes emplear el comando SQR para muchas otras aplicaciones, aparte de hallar el lado de un cuadrado cuya área conoces. Hay muchas ecuaciones matemáticas que contienen cuadrados y raíces cuadradas; en tales casos puedes utilizar la función SQR para resolverlos con ayuda de tu ordenador.

Como ejemplo, en el siguiente programa se emplea una ecuación para calcular el tiempo que tardará en estrellarse contra las rocas un objeto que cae desde un acantilado, o cualquier otra altura (sin tener en cuenta la resistencia del aire). También podrás saber la velocidad con que se está moviendo el objeto en el momento en que llega al suelo. El porqué de la relación existente entre los cuerpos que caen y los cuadrados y las raíces cuadradas reside en que todo lo que está bajo la influencia de la gravedad, cae con mayor velocidad cuanto más tiempo lleva cayendo (siempre sin tener en cuenta la resistencia del aire). De hecho, el camino recorrido está relacionado con el cuadrado del tiempo. Pero, antes de mirar esto con más detalle, ensayemos el programa:

```
1Ø CLS
2Ø INPUT "INTRODUCE LA
   DISTANCIA DE CAIDA
   (METROS)",D
3Ø IF D<Ø THEN GO TO 2Ø
4Ø LET T=SQR ((2*D)/9.81)
5Ø LET V=SQR (2*D*9.81)
6Ø LET T=INT (T*1ØØ)/1ØØ
7Ø LET V=INT (V*1ØØ)/1ØØ
8Ø PRINT  INVERSE
   1´´"TIEMPO QUE TARDA EN
   LLEGAR  AL SUELO:";
   INVERSE Ø´T;" SEGUNDOS"
9Ø PRINT  INVERSE 1´"MAXIMA
   VELOCIDAD:"; INVERSE Ø´V;"
   METROS POR SEGUNDO"
1ØØ PRINT "(";INT (2.25*V+.5);
    " MPH)"
2ØØ PRINT ´´"PRESIONA UNA
    TECLA PARA JUGAR DE
    NUEVO"
21Ø IF INKEY$="" THEN GO TO
    21Ø
22Ø GO TO 1Ø
```

El ordenador empieza borrando la pantalla. Después te indicará que teclees la altura desde la que quieras calcular la caída. Si tecleas un número negativo, el ordenador te pedirá que teclees un nuevo número, ya que no se puede calcular la raíz cuadrada de un número negativo.

A continuación el ordenador calcula el tiempo necesario para que el cuerpo que cae llegue hasta el suelo, y la velocidad que lleva en el momento del impacto. Esto se hace en las líneas 40 y 50.

La ecuación utilizada en la línea 40 es una versión de una de las «ecuaciones del movimiento». La hemos reordenado para que adopte la siguiente forma:

$$T = \sqrt{(2*D)/a}$$

donde **T** es el tiempo requerido para recorrer una distancia dada (en este caso para llegar hasta el suelo), **D** es la distancia que tú introduces, y **a** es la aceleración, que es una medida de lo que va cambiando la velocidad en cada segundo.

En el programa de ordenador no interviene ninguna variable que se llame **a**, debido a que su valor no cambia. Como puedes ver, figura el valor 9.81 en lugar de **a**.

La ecuación anterior se presenta resuelta en la línea 40. El resultado es el tiempo que el objeto tardará en chocar con la tierra.

En la siguiente línea se resuelve una ecuación análoga para calcular la velocidad con que se mueve el objeto en el momento de su impacto con el suelo.

$$V = \sqrt{2*D*9.81}$$

En esta ecuación, en vez de dividir el número 2*D por la aceleración, el ordenador multiplica dicho número por la aceleración. La V significa velocidad.

En ambas ecuaciones, el símbolo $\sqrt{\ }$ cubriendo a 2*D*9.81 o a 2*D/9.81 significa «raíz cuadrada de». Aquí es, pues, donde se utiliza la función raíz cuadrada en este ejemplo. Una vez que tengas la respuesta a estas ecuaciones, puedes modificarlas ligeramente para hacer que el ordenador calcule la altura desde la que se arrojó el objeto.

En su forma general, la ecuación se aplica a cualquier objeto que se ve sometido a aceleración, desde un ladrillo que cae, hasta un cohete lanzado desde la Luna. Lo que hace falta para relacionarlo específicamente con un objeto que cae es el valor correcto de la aceleración. En este caso la aceleración es la debida a la gravedad, por lo que a la variable **a** se le ha asignado este valor. La gravedad tiene una aceleración de 9.81 metros por segundo y por segundo. En otras palabras, para un objeto que cae, su velocidad aumenta en cada segundo 9.81 metros por segundo. En vez de metros por segundo y por segundo, también se puede decir metros por segundo al cuadrado, por lo que ya te puedes imaginar por dónde aparece la función de elevación al cuadrado.

Las ecuaciones para hacer esto serían algo así:

$$D = \frac{a*(t \uparrow 2)}{2}$$

o bien

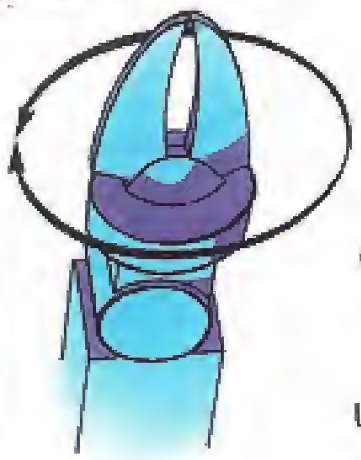
$$D = V \uparrow 2/(a*2)$$

Ensaya a cambiar tu programa de ordenador para calcular las respuestas por medio de estas dos ecuaciones. En vez de introducir la altura, podrías introducir la velocidad a la que quisieras que cayera algo sobre la tierra (en el caso de la segunda ecuación) o el tiempo que quieres que transcurra antes de llegar al suelo. En ambos casos, tu ordenador calculará la altura necesaria para satisfacer tu petición.

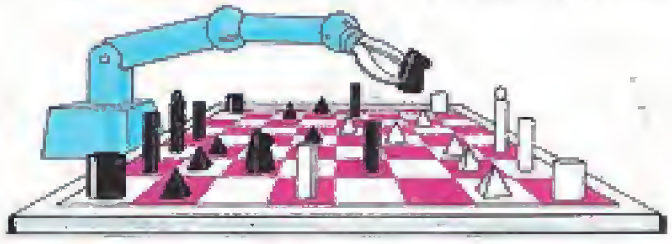
La posibilidad de resolver problemas de esta clase tiene gran cantidad de aplicaciones prácticas, si bien normalmente las ecuaciones deben considerar otras influencias que también actúan sobre el cuerpo que se mueve. Pero podrías calcular por ejemplo los detalles del comportamiento de un coche que se acelera en una prueba de frenos, o bien reconstituir lo que sucedió en un determinado accidente. En este último caso, puede que quisieras conocer la velocidad a que se estaba desplazando el vehículo en el instante en que chocó. Es posible calcular esto, haciendo la cuenta atrás a partir de la distancia que viajó después del momento del impacto, siempre que dispongas de los datos adecuados para introducirlos en tu ecuación.

Las modelizaciones que te permite el uso de funciones potenciales no se agotan en cosas tales como el tamaño de tu casa o el comportamiento de tu automóvil. Se aplican también a otros cálculos muy distintos, tales como el crecimiento de un árbol o el porqué un ave que fuera tan grande como un elefante jamás podría levantarse del suelo. En un próximo artículo veremos cómo hacer aún un uso todavía más eficaz de las funciones matemáticas de tu ordenador, incluida la forma de conseguir algunos efectos gráficos realmente impresionantes.

¡PARTICIPA EN EL CONCURSO!



En INPUT estamos convencidos de que aún puedes hacer muchas más cosas con tu ordenador. Sin duda, muchos lectores estareis utilizando vuestro micro para funciones de lo más variadas, en unos casos; pintorescas, en otros; mientras que algunos listillos habrán podido utilizarlo para resolver tareas complejas. Es lógico, modificando programas y variando los periféricos nuestro ordenador puede prestar sus servicios en infinidad de facetas. INPUT quiere que esas aplicaciones y utilidades a las que has conseguido dedicar tu ordenador, sean conocidas por todos sus lectores y por eso ha organizado el «Concurso de Aplicaciones y Utilidades», en el que puede participar cualquiera de nuestros lectores.



BASES

UTILIDADES Y APLICACIONES: Si tu ordenador controla la calefacción de tu casa, gobierna un robot, dirige un pequeño negocio, organiza la maqueta de tu tren eléctrico, o cualquier cosa interesante u original; envíanos información gráfica y listados de tus programas, grabados en un cassette, diskette o microdrive.

Todo ello habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvideis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante los próximos tres meses SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas. en material microinformático a elegir por los afortunados.**

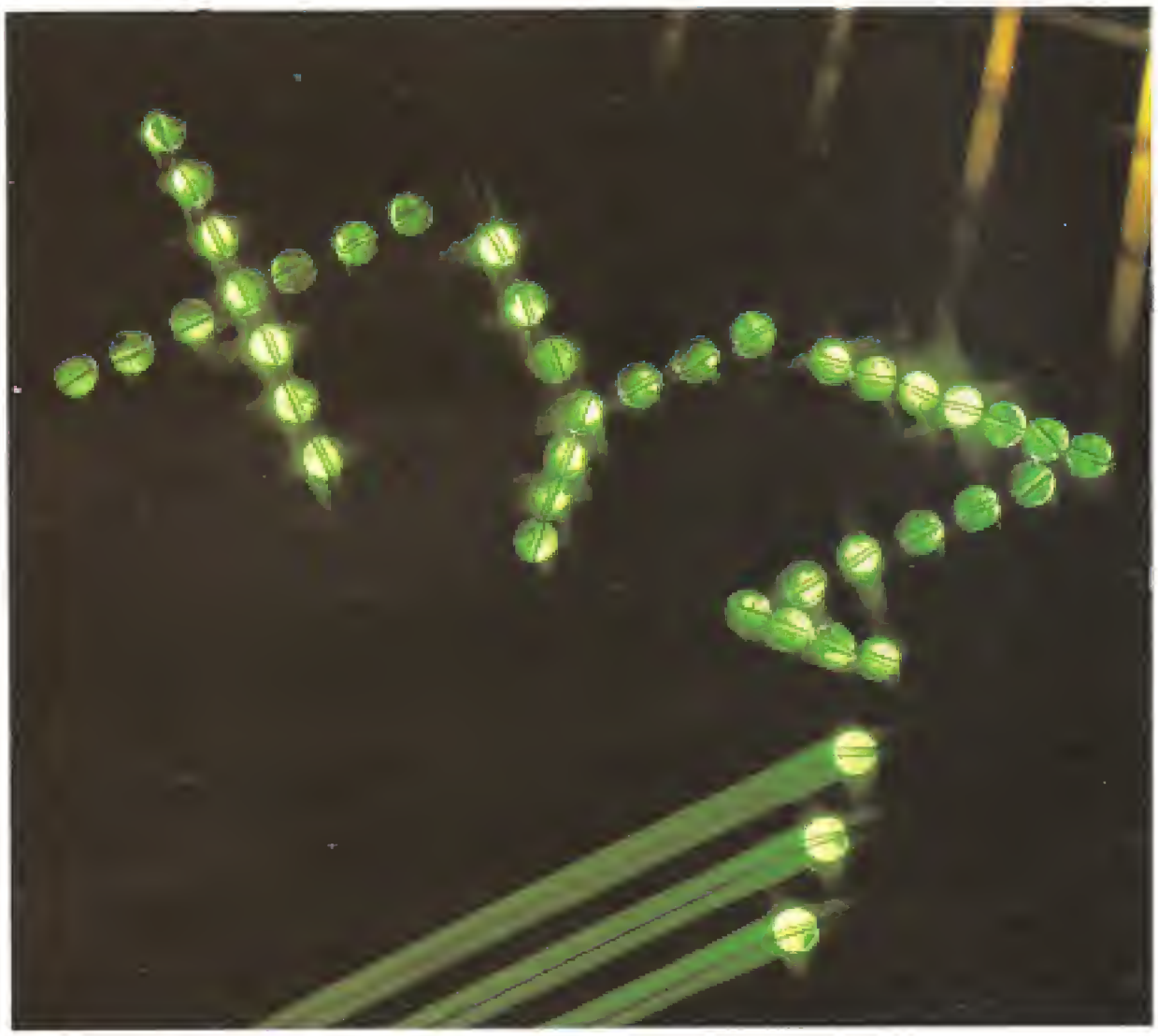
¡No os desanimeis!, por muy simples o complejas que puedan parecer vuestras ideas, todas están revisadas con el máximo interés.

INPUT SINCLAIR
Aribau, 185. Planta 1.ª
08021 BARCELONA

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.

MAS IDEAS PARA TUS TITULARES

- DIBUJO DE LETRAS LINEA A LINEA
- CAMBIO DE LA ESCALA DE LOS CARACTERES
- INTRODUCCION DE LAS PALABRAS



En la primera parte de este artículo vimos la forma de crear letras en pantalla empleando el conjunto de caracteres gráficos. Aquí tienes nuevo material tipográfico más unas cuantas ideas sobre la utilización de todas las nuevas letras.

Los modelos tipográficos que puedes crear con ayuda de los programas que te hemos proporcionado se construyen a partir de los caracteres gráficos, basados en el conjunto de caracteres estándar, o en los bloques gráficos. A continuación te presentamos una nueva forma de crear letras para hacer presentaciones en pantalla, utilizando los gráficos de alta resolución para dibujarlas línea a línea con ayuda de la sentencia DRAW.

Los programas que siguen funcionan de una manera muy parecida a la de los bloques gráficos de letras, en el sentido de que cada uno de ellos consiste en una serie de sentencias DATA que indican al ordenador de qué manera tiene que construir cada letra. Esta información se almacena en un *array*, y tú puedes introducir las palabras que quieras ensanchar en forma de cadena de caracteres. Igual que en aquel caso, las letras se van construyendo con arreglo a las instrucciones que el ordenador va encontrando, y se van representando en pantalla. Posteriormente verás de qué manera puedes utilizar esta presentación (o la que proceda de cualquier otro generador de letras) como parte de tus propios programas.

El uso de líneas con alta resolución te permite una libertad mucho mayor en el estilo de tus letras que los métodos anteriores. La ampliación del conjunto de caracteres sólo está limitada por el doble de la altura y de la anchura de los caracteres estándar. Las letras de los bloques gráficos están completamente fijadas por su diseño en las sentencias DATA.

Las sentencias DATA de los siguientes programas también fijan de qué manera se construye cada letra (así, por ejemplo, para una L, el ordenador recibe instrucciones para trazar una línea vertical y otra horizontal). Pero todas estas instrucciones están relacionadas unas con otras, por lo que no determinan el aspecto global de la letra. Piensa como si cada letra estuviera encerrada en una caja imaginaria. Si dicha caja es alta y estrecha, obtendrás una letra alta y estrecha, mientras que si es baja y ancha, la letra resultante será baja y ancha. Los programas están escritos de tal forma que puedes decidir estos factores de escala al principio.

Teclea ya las líneas del programa.

```
1Ø POKE 23658,8
2Ø DIM N(26): DIM A(26,12,2)
3Ø FOR N=1 TO 26
4Ø READ N(N)
5Ø FOR M=1 TO N(N)
6Ø READ A(N,M,1),A(N,M,2)
7Ø NEXT M
8Ø NEXT N
1ØØ INPUT "METE UN STRING",
    LINE A$: IF A$="" THEN
    GO TO 1ØØ
11Ø INPUT "METE FACTOR-X",X
12Ø INPUT "METE FACTOR-Y",Y
125 CLS
13Ø FOR N=1 TO LEN A$
14Ø LET T$=A$(N): IF T$<"A"
    OR T$>"Z" THEN NEXT N:
    GO TO 1ØØ
15Ø PLOT
```

```
    (1Ø*(N-1)*X)+X*A(CODE
    T$-64,1,1),2Ø+Y*A(CODE
    T$-64,1,2)
16Ø FOR M=2 TO N(CODE
    T$-64)
17Ø DRAW X*A(CODE T$-64,M,
    1),Y*A(CODE T$-64,M,2)
18Ø NEXT M
19Ø NEXT N
2ØØ GO TO 1ØØ
1ØØØ DATA 8,Ø,Ø,Ø,5,1,1,4,Ø,
     1,-1,Ø,-5,Ø,3,-6,Ø
1Ø1Ø DATA 12,Ø,Ø,Ø,6,5,Ø,1,
     -1,Ø,-1,-1,-1,-5,Ø,5,
     Ø,1,-1,Ø,-1,-1,-1,-5,
1Ø2Ø DATA 8,6,1,-1,-1,
     -4,Ø,-1,1,Ø,4,1,1,4,Ø,
     1,-1
1Ø3Ø DATA 7,Ø,Ø,Ø,6,4,Ø,2,
     -2,Ø,-2,-2,-2,-4,Ø
1Ø4Ø DATA 7,6,Ø,-6,Ø,Ø,6,6,
     Ø,-6,Ø,Ø,-3,5,Ø
1Ø5Ø DATA 6,Ø,Ø,Ø,6,6,Ø,-6,
     Ø,Ø,-3,5,Ø
1Ø6Ø DATA 1Ø,5,2,1,Ø,Ø,-1,
     -1,-1,-4,Ø,-1,1,Ø,4,1,
     1,4,Ø,1,-1
1Ø7Ø DATA 6,Ø,Ø,Ø,6,Ø,-3,6,
     Ø,Ø,3,Ø,-6
1Ø8Ø DATA 6,Ø,Ø,6,Ø,-3,Ø,Ø,
     6,-3,Ø,6,Ø
1Ø9Ø DATA 5,Ø,1,1,-1,4,Ø,1,
     1,Ø,5
11ØØ DATA 6,Ø,Ø,Ø,6,Ø,-3,6,
     3,-6,-3,6,-3
111Ø DATA 3,6,Ø,-6,Ø,Ø,6
112Ø DATA 5,Ø,Ø,Ø,6,3,-3,3,
     3,Ø,-6
113Ø DATA 4,Ø,Ø,Ø,6,6,-6,Ø,6
114Ø DATA 9,1,Ø,-1,1,Ø,4,1,
     1,4,Ø,1,-1,Ø,-4,-1,-1,
     -4,Ø
115Ø DATA 7,Ø,Ø,Ø,6,5,Ø,1,
     -1,Ø,-1,-1,-1,-5,Ø
116Ø DATA 9,4,2,2,-2,-5,Ø,
     -1,1,Ø,4,1,1,4,Ø,1,-1,
     Ø,-5
117Ø DATA 9,Ø,Ø,Ø,6,5,Ø,1,
     -1,Ø,-1,-1,-1,-5,Ø,4,
     Ø,2,-3
118Ø DATA 12,Ø,1,1,-1,4,Ø,1,
     1,Ø,1,-1,1,-4,Ø,-1,1,Ø,
     1,1,1,4,Ø,1,-1
119Ø DATA 4,3,Ø,Ø,6,-3,Ø,6,Ø
12ØØ DATA 6,Ø,6,Ø,-5,1,-1,
     4,Ø,1,1,Ø,5
121Ø DATA 3,Ø,6,3,-6,3,6
122Ø DATA 5,Ø,6,1,-6,2,3,2,
     -3,1,6
123Ø DATA 5,Ø,Ø,6,6,-3,-3,
     -3,3,6,-6
124Ø DATA 5,3,Ø,Ø,3,-3,3,3,
     -3,3,3
125Ø DATA 4,6,Ø,-6,Ø,6,6,
     -6,Ø
```

Como puedes ver, el programa contiene 26 sentencias DATA, una para cada una de las letras del alfabeto. Y cada una de dichas sentencias contiene una serie de números que indican al ordenador cómo tiene que dibujar la forma de la letra en cuestión, según una serie de trazos cortos. El primer número que figura a continuación de cada DATA es el número total de trazos que se utilizan para formar cada letra. Por ejemplo la L necesita menos que la S. El mayor número utilizado es el 12.

Los siguientes números están ordenados por parejas que dan las coordenadas *x* e *y* de cada uno de los segmentos. Tal como hemos explicado anteriormente, se trata de números relativos y no absolutos, por lo que las líneas que se dibujan realmente pueden ir afectadas por factores de escala. El primer par de coordenadas especifica un punto de partida para la letra dentro de su «caja» imaginaria.

Los números contenidos en las sentencias DATA se leen por medio de las correspondientes READ y se pasan a dos *arrays* llamados N y A. A es un *array* tridimensional de 26 × 12 × 2 dimensiones, que corresponden respectivamente al número de letras, el número máximo de líneas y los vectores *x* e *y*.

Las líneas que te permiten introducir tus propias palabras realizan una comprobación para asegurarse de que en efecto has entrado una cadena de caracteres y no una cadena «nula». Por lo demás, no hay límite en cuanto a la longitud de la cadena a introducir.

Los programas te permiten entrar

1. Aquí tienes tres versiones de los nuevos caracteres tipográficos del Spectrum.

dos valores: un «factor X» y un «factor Y». Los valores que introduzcas aquí determinan el tamaño real de cada letra (es decir, las dimensiones de la «caja» invisible). Para que te hagas una idea en cuanto a la escala, el número 1 corresponde a un carácter de tamaño estándar. Con el 2 obtendrías doble anchura o doble altura, mientras que el 0.5 daría como resultado la mitad de la anchura o altura. Observa que con valores más pequeños que 1 puedes conseguir algunos efectos inesperados. Como el ordenador no puede dibujar una fracción de pixel, trazará por defecto un doble pixel. En consecuencia, las letras que contengan un gran número de líneas tienen más posibilidades de que les suceda una cosa de este tipo. Como resultado podría ocurrir, por ejemplo, que la S resultara ligeramente superior a la T.

Al disponer de valores diferentes para cada uno de los factores, estás en condiciones de producir algunos efectos interesantes para variar los caracteres: ya sea con caracteres altos y estrechos o bajos y cortos.

Como no existe un límite para la longitud de la cadena que puedes introducir, tienes que tener cuidado de no teclear demasiadas letras para que el espacio disponible en pantalla para la representación de las mismas resulte suficiente. De no ser así, el ordenador se detendrá y te enviará un mensaje de error o producirá extrañas variaciones en las letras.

Puedes calcular de forma muy sencilla el número de caracteres que te caben para un determinado factor de es-

2. Modificando los factores de escala de X e Y, las letras pueden adquirir cualquier tamaño.

3. Estas imágenes son ejemplos de letras normales, extra-anchas y extra-altas.

cala. Observa que el valor que importa es el factor X, ya que es el que define cuánto más ancha o más estrecha es cada letra (el factor Y determina cómo es la letra de alta). Si tienes un valor de 2 como factor de multiplicación, sólo podrás acomodar la mitad de caracteres de los que te cabían en una línea normal. Si en cambio tuvieras un factor de multiplicación de 4, únicamente podrías poner la cuarta parte de caracteres.

Al igual que otros programas para crear una determinada tipografía, éste utiliza un bucle principal para recorrer cada letra de la cadena. Dicho bucle empieza en la línea 130.

El programa prosigue de forma análoga a la del anterior generador de letras, definiendo una variable de tipo cadena de caracteres (T$) que hace igual a cada letra. A continuación examina si dicho carácter es una letra de la A a la Z (cualquier otro carácter se trata como si fuera un espacio). De no ocurrir esto, el ordenador ejecuta un NEXT para actualizar el bucle FOR ... NEXT, y si ya no hay más letras en la cadena de caracteres el ordenador te permite introducir una nueva cadena.

Cuando el ordenador encuentra una letra dentro de la cadena, se va a la línea 150 en la que comienzan las rutinas de dibujo.

La variable de control del bucle FOR ... NEXT, N, se utiliza como guía para la coordenada *x* de la posición de partida a la cual el ordenador le añade dos cosas. La primera es un valor relativo, las coordenadas de dibujo tomadas del *array* A, mientras que la segunda es un valor para tener en cuenta el tamaño de la pantalla y tus factores de escala *x* e *y*.

Ya has visto de qué forma uno de los tres elementos de este *array* se emplea para separar los detalles de las coordenadas *x* e *y*.

Los otros dos elementos asignan a cada una de las 26 letras hasta 24 números; recuerda que el máximo número de segmentos utilizados en el dibujo de una letra es 12. El *array* unidimensional N se emplea para informar al ordenador de cuántas líneas tiene cada letra. Así, por ejemplo, cuando el ordenador tiene que dibujar la letra A, primero mira en el *array* N cuántas líneas tiene que dibujar. Este número controla, entonces, el bucle para dibujar el número exacto de líneas para cada letra.

Los valores de los números de A, el *array* principal, de tres dimensiones, son 64 menos el código de carácter correspondiente a cada letra; ésta es la razón de que aparezca la expresión CODE T$−64 en las líneas 130 a 180. La razón de esto es que el ordenador puede así tomar el código de carácter de la letra y utilizarlo para la cuenta en la parte derecha de los *arrays* al dibujar las líneas. Los valores relativos de *x* e *y* almacenados en el *array* se multiplican entonces por el factor de escala que hayas introducido y pasan a los comandos DRAW.

Cuando el ordenador termina de dibujar una letra, pasa a la siguiente letra de la cadena de caracteres, suponiendo que haya una siguiente letra, y repite todo el proceso para dibujarla. Si ya no hay más letras que dibujar, el ordenador regresa a la línea 100 para que puedas introducir una nueva cadena de caracteres.

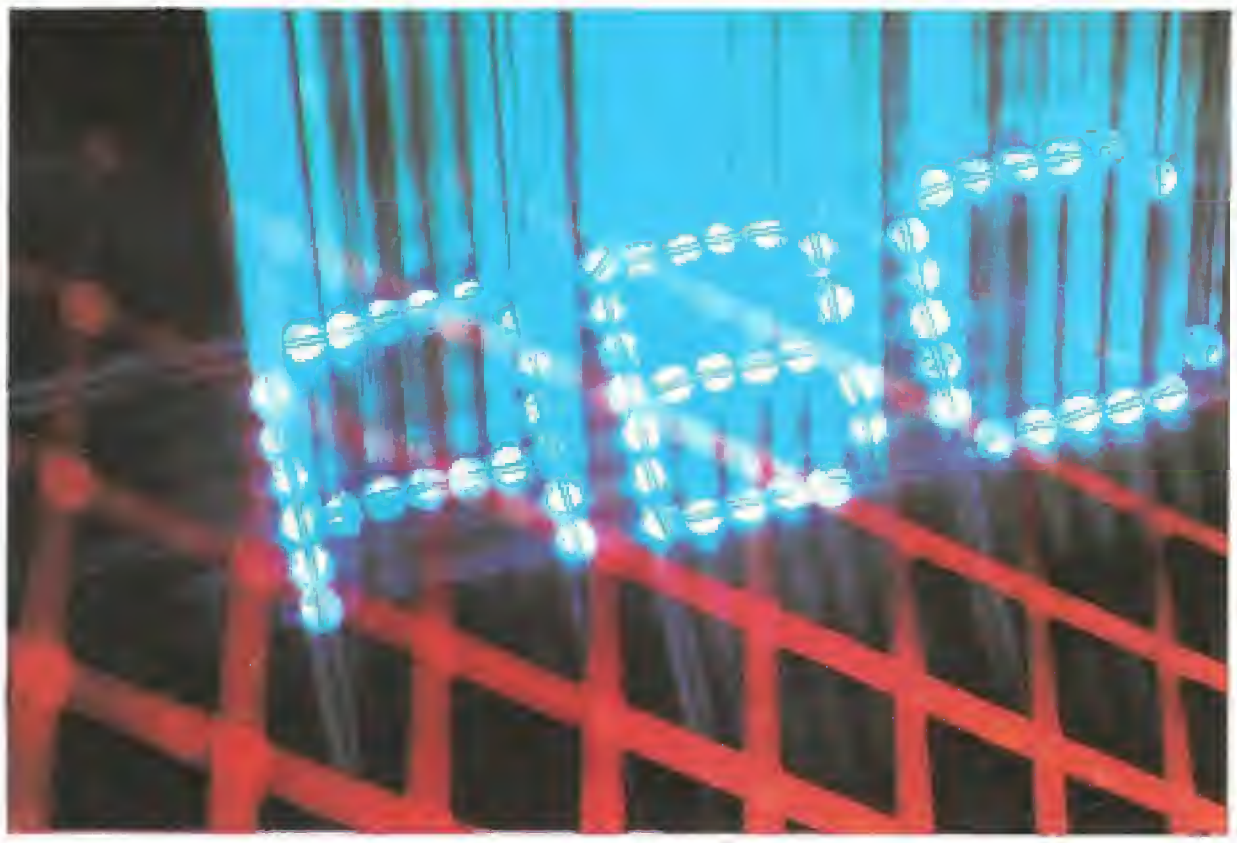
Ya conoces tres ejemplos de nuevos caracteres tipográficos, pero puedes aplicar las mismas ideas para crear tus propias letras especiales. Aunque no puedes modificar la doble anchura o altura de los caracteres gráficos propios del ordenador, puedes utilizar, sin embargo, las rutinas de «expansión» para ampliar tus propios caracteres redefinidos.

Las variantes que puedes introducir son casi ilimitadas: puedes diseñar tus propios caracteres a partir de los gráficos de tu ordenador y almacenarlos en un *array*. Si no estás satisfecho con los caracteres que lleva incorporados tu ordenador, no hay razón para que no puedas diseñar tus propios GDUs y combinarlos. También podrías construir nuevas variantes de los modelos tipográficos que hemos visto en este artículo; para ello te basta con modificar los valores de los DATA una vez que tengas definidos los nuevos caracteres, y cambiar el *array* en caso de que sea necesario.

Lo de crear letras grandes está muy bien, pero no tendría sentido a menos que puedas utilizarlas en tus programas. Si empleas el programa de dibujo de este artículo, tienes que poder llamar a las letras especiales para poder aplicarlas en cualquier parte. Hay varias formas de utilizar los caracteres tipográficos especiales en tus propios programas. La solución más evidente consiste en incorporar los programas generadores de letras en forma de subrutina y después poner un GOSUB cada vez que quieras acceder a ella.

Si quieres emplear la rutina de expansión del conjunto de caracteres, ésta es, probablemente, la mejor solución. Sin embargo, este programa de dibujo de letras que acabamos de ver, dista bastante de ser ideal. La principal razón de esto es que ocupa una gran cantidad de espacio de memoria, con lo que no te quedará mucho disponible para tu propio uso.

Puedes solucionar este problema de diversas formas, todas las cuales im-



plican la utilización de programas para dibujar tus propias letras y el almacenamiento de los resultados de forma que queden disponibles para su uso posterior donde haga falta.

Una manera de hacer esto es almacenar tus diseños en GDUs, para lo cual puedes «pokear» el correspondiente GDU con el contenido de la pantalla.

Otra alternativa es almacenar en cinta con un SAVE la sección de memoria correspondiente a la pantalla como bloque en código máquina. Posteriormente puedes volver a hacer una carga con LOAD del gráfico desde la cinta al ordenador. Mientras dicho gráfico esté en el ordenador, tienes que protegerlo almacenándolo por encima del extremo del BASIC. Una vez que el gráfico esté en la memoria, ya estás en condiciones de utilizarlo.

La mejor manera de hacerlo consiste en escribir una rutina para extraer mediante PEEKs el correspondiente código contenido en la zona protegida de la RAM y a continuación «pokear» dicho código en la zona de memoria correspondiente a la pantalla, o modificar el conjunto de punteros de caracteres para que tu código «se convierta» en varios caracteres. Seguidamente podrías imprimir los caracteres para hacer que aparezcan tus letras grandes. La rutina para desplazar tu bloque de código desde su área protegida en RAM hasta la pantalla, sería más o menos así:

```
1ØØØ FOR x = 1 TO (longitud de
     tu bloque de codigo)
1Ø1Ø POKE (direccion inicial de
     la memoria de pantalla) +
     x, PEEK (direccion inicial
     de tu bloque de codigo) + x
1Ø2Ø NEXT x
```

Tampoco en este caso es esta rutina una solución especialmente acertada, ya que tardaría bastante en terminar, a menos que conozcas el código máquina y puedas escribir una rutina equivalente en este lenguaje. En tal caso, podrías llamar a dicha rutina de forma que tus letras aparecerían en la pantalla de forma casi instantánea.

La otra alternativa que tienes es almacenar mediante un SAVE la imagen que tengas en pantalla y cargarla con un LOAD cuando hagas la carga de tu programa. La imagen permanecerá entonces en pantalla hasta que la elimines al hacer un borrado de pantalla.

El Spectrum dispone de una palabra clave especial que te permite almacenar en cinta el contenido de la pantalla de forma muy sencilla. La imagen se almacena como un bloque de memoria, con la salvedad de que en vez de tener que teclear la dirección de comienzo de la pantalla y la cantidad de memoria que quieres almacenar, te limitas a teclear la palabra clave SCREEN$ inmediatamente después del nombre del fichero. De esta forma, con el siguiente comando directo se almacenaría la imagen llamada «pic»:

```
SAVE"pic"SCREEN$
```

Para volver a cargar la imagen de pantalla, teclea lo siguiente:

```
LOAD""SCREEN$
```

Puedes utilizar una imagen como página de entrada con el título de un juego, haciendo que se cargue al principio y permanezca en pantalla mientras se va cargando el juego. Para ello tienes que escribir un programa cargador:

```
1Ø LOAD""SCREEN$
2Ø LOAD"GAME"
```

y a continuación almacenarlo por medio de:

```
SAVE"LOADER"LINE1Ø
```

De esta forma se ejecuta automáticamente. Tienes que almacenar, además, el propio juego tecleando SAVE «GAME» LINE 10, para que también éste se ejecute automáticamente. En la cinta, los programas han de ir almacenados en el orden correcto: LOADER (cargador), SCREEN (pantalla) y GAME (juego).

Los tipos de letras grandes que hemos visto hasta ahora no son en absoluto las únicas clases de letras grandes que puedes crear con tu ordenador. Ya has visto cómo una de las maneras de almacenar las letras consiste en utilizar los GDUs. También puedes poner varios GDUs juntos, combinándolos para formar una letra más grande. Por medio de este método puedes diseñar GDUs tan complicados como quieras, ya que el ordenador sólo tiene que hacer un PRINT de una cadena de caracteres, no teniendo que dibujar, en realidad, cada carácter, como hacen los otros programas.

BETA BASIC 3.0: LA SUPERACION

- UN ALTO GRADO DE PERFECCION
- ANALISIS DEL MANUAL
- FACILIDADES GRAFICAS
- PROCEDIMIENTOS

La tercera versión de este programa es realmente una superación. La longitud y complejidad han crecido en cierto grado, pero se ha conseguido un alto grado de perfeción. El cometido que se pretendía, conseguir un ampliador del BASIC del Spectrum, ha sido alcanzado con creces aunque siempre queda algo por realizar, otra nueva versión...

Antes de adentrarnos en el «mundo» del programa, hemos de armarnos de paciencia e intentar desmenuzar poco a poco el complicado entramado que supone el manual, nada menos que 87 páginas.

Quizá el fabricante, previendo esta posible desesperación, hace notar este punto y adjunta una breve introducción de las facilidades que aporta el programa.

En relación a la versión 1.8, el programa ha incluido un gran número de nuevos comandos y funciones, pero sobre todo ha potenciado los ya existentes, con lo que la longitud del 3.0 se agranda hasta los 18 K, situando el RAMTOP en la posición 47070. La estructura del programa sigue siendo la misma, el código máquina por un lado y las tres líneas BASIC aparte. Dichas líneas nos ofrecen la posibilidad de pasar el programa a disco o microdrive y la línea 0 (definición de las funciones) es la clave para conseguir la adaptación de las rutinas realizadas con otras versiones.

Se ha conseguido una gran compatibilidad ascendente: una vez cambiada la línea 0 al comienzo de cada rutina, los programas hechos con la 1.8 ruedan perfectamente con el 3.0 (pero no todos los hechos con el 3.0 funcionan con la versión 1.8). Como prueba de esta perfecta compatibilidad sólo resaltar que la única variable utilizada en el tratamiento de errores llamada «LINE», se ha cambiado por «lino» para evitar así una posible confusión con el comando LINE (INPUT LINE, SAVE LINE).

Aunque se siguen manteniendo todas las facilidades anteriores (movimiento del cursor, acelerador de programas, etc.), sin duda alguna los puntos fuertes de esta nueva adaptación son los que enumeramos y comentamos a continuación:

— Los distintos formatos de listado que se pueden conseguir, tanto en pantalla como en impresora, lo cual nos ayudará mucho en nuestras largas sesiones de programación.

— Se han ampliado las posibilidades de KEYWORDS, por lo que con sus nuevas opciones podremos incluso teclear letra a letra nuestros comandos, olvidándonos de su situación en el teclado y asemejándose más a otros ordenadores (de un plumazo se echa por tierra uno de los principales argumentos de los detractores del Spectrum). Con esta forma de trabajar y funciones como SHIFT$ se evitan también las posibles dificultades a la hora de listar los nuevos comandos en la impresora (problema existente con la versión 1.8).

— Todo el tema concerniente al manejo de procedimientos ha sido grandemente mejorado, tanto en posibilidades como en sencillez de manejo (ahora podemos recurrir a un procedimiento con sólo teclear su nombre, para lo cual podemos abandonar el modo K y pasar a L o C sin introducir un comando previo).

— Se ha potenciado mucho el manejo de todo tipo de variables y matrices (podemos trasladar parte de una matriz, modificarla, rellenarla, ordenarla...).

— A nivel gráfico podemos incluso llegar a tratar con partes diferenciadas de la pantalla que previamente podemos dividir en sectores o ventanas. Podemos seleccionar la escala de los gráficos o caracteres a imprimir, almacenar partes de la pantalla a modo de variables, etc.

— Se han facilitado las cargas y grabaciones de programas y datos, sobre todo en el Microdrive.

A continuación reseñamos las funciones y comandos más interesantes que añade la versión 3.0 pero sin repetir los ya comentados para la 1.8 (pues son exactamente los mismos).

COMANDOS

Antes de empezar a programar es necesario elegir el modo en el que pretendemos trabajar. Como ya hemos comentado, a parte de los modos KEYWORDS 1 o 0, ahora podemos seleccionar 3 modos más:

— KEYWORDS 2: Modo normal en el que los comandos del *Beta Basic* son obtenidos pulsando la tecla correspondiente en modo gráfico.

KEYWORDS 3: Es la opción utilizada en principio por el *Beta Basic*. Además de funcionar como el modo KEYWORDS 2, podemos también si queremos introducir los comandos tecleándolos letra a letra (igual los comandos del Beta BASIC como los del Spectrum). Para conseguir abandonar el modo K deberemos dejar un espacio antes de la primera tecla pulsada.

— KEYWORDS 4: Funciona como el modo anterior pudiendo introducir los comandos de las dos formas, pero con la salvedad de que no estaremos nunca en el modo K a no ser que pulsemos las teclas Symbol Shift + Enter. Ésta es la manera más parecida de trabajar a la de otros ordenadores distintos del Spectrum. La única posible limitación de los dos últimos modos de trabajo es que los nombres de las variables o procedimientos no han de coincidir con ninguno de los comandos.

FACILIDADES GRAFICAS

* WINDOW n (x, y, a, l):
Como su traducción al castellano indica, este comando nos permite crear una serie de «ventanas» en nuestra pantalla que serán totalmente independientes unas de otras en cuanto a su posición de impresión y atributos. «n» es el número de referencia de cada ventana y WINDOW 0 es la pantalla normal. La esquina superior izquierda de la ventana viene indicada por las coordenadas «x», «y» y su anchura y longitud por «a» y «l». Una vez especificados estos datos, cada vez que tecleamos «WINDOW n» estamos haciendo referencia a esa ventana, y todo comando que a continuación altere algunos de los atributos o imprima algo, sólo afectará a esa parte de la pantalla. Todos los datos referidos a cada una de las ventanas están protegidos por el RAMTOP (a salvo de NEW).
Ej.:

```
 5 BORDER Ø: INK 9: CLS
1Ø FOR n=1 TO 4
2Ø WINDOW n,Ø+(64*(n-1)),
   175,64,176
3Ø NEXT n
4Ø FOR f=1 TO 4: WINDOW f:
   PAPER f: CLS: NEXT f
5Ø FOR g=1 TO 4: WINDOW g:
   LIST g: NEXT g
6Ø CLS 1: CLS 2: REM borra las
   ventanas 1 y 2
7Ø FOR n=1 TO 1Ø: PRINT
   WINDOW 2; PAPER 7; INK 1;
   BRIGHT 1; «INPUT»: PRINT
   WINDOW 2; PAPER 1; INK 7;
   BRIGHT 1; «SINCLAIR»:
   NEXT n
8Ø GET a$: PRINT WINDOW 1;
   INK 7; BRIGHT 1; a$;: GO TO
   8Ø
```

El programa muestra cómo hemos creado 4 «ventanas» a modo de franjas verticales en la pantalla, y en cada una de estas zonas generamos un listado independiente que se verá limitado por el número de columnas de cada ventana. El color elegido en cada una es ajeno al de las demás, y puede alterarse fácilmente con sólo hacer referencia a la WINDOW n elegida. La segunda parte del programa muestra cómo imprimir «INPUT SINCLAIR» en la ventana 2, escogiendo de manera fácil los atributos, número de veces de impresión, etc..., y además simula una máquina de escribir en WINDOW 1, resaltando aún más el tema de la limitación de columnas en esa ventana.

CSIZE a (,l):
Controla el tamaño de los caracteres que van a ser utilizados, especificando su anchura «a» y altura «l» (en pixel). Si no indicamos la altura se supone a=1. El juego de caracteres normales utiliza un CSIZE 8. Así CSIZE 24 generará una serie de caracteres 3 veces más grandes. CSIZE 16,8 producirá caracteres el doble de anchos. Aplicando este comando a una ventana determinada, podemos conseguir distintos tipos de caracteres para cada zona de la pantalla.
Ej.: El comando CSIZE puede ser utilizado en modo general: CSIZE 32: LIST o CSIZE 32,8: LIST, o también dentro del ámbito de un comando como PRINT:

```
1Ø PRINT CSIZE 16,32; PAPER
   6; «Input Sinclair»; PRINT:
   PRINT: PRINT CSIZE 32,16;
   PAPER 1; INK 7; BRIGHT 1;
   «Sinclair»
```

* CLS n:
Sigue funcionando como estamos acostumbrados, pero si fijamos «n» borrará la ventana especificada.

* GET a$,x,y (,a,l)(;t):
Aparte del otro uso explicado al hablar de la versión 1.8, se han aumentado sus funciones. Ahora el comando puede también adquirir esta otra forma para trabajar como instrucción gráfica. Su utilización nos permite almacenar una parte de pantalla en la variable elegida (no tiene por qué llamarse a$) especificando la esquina superior izquierda del bloque a almacenar (x,y) y su anchura y altura en bytes o bloques de 8 por 8 pixel (a,l). Si no se indican estos dos últimos valores se asume que el gráfico a guardar es del tamaño de un carácter. Los gráficos almacenados son precedidos por un código de control para indicar el tipo de los datos almacenados en la variable.

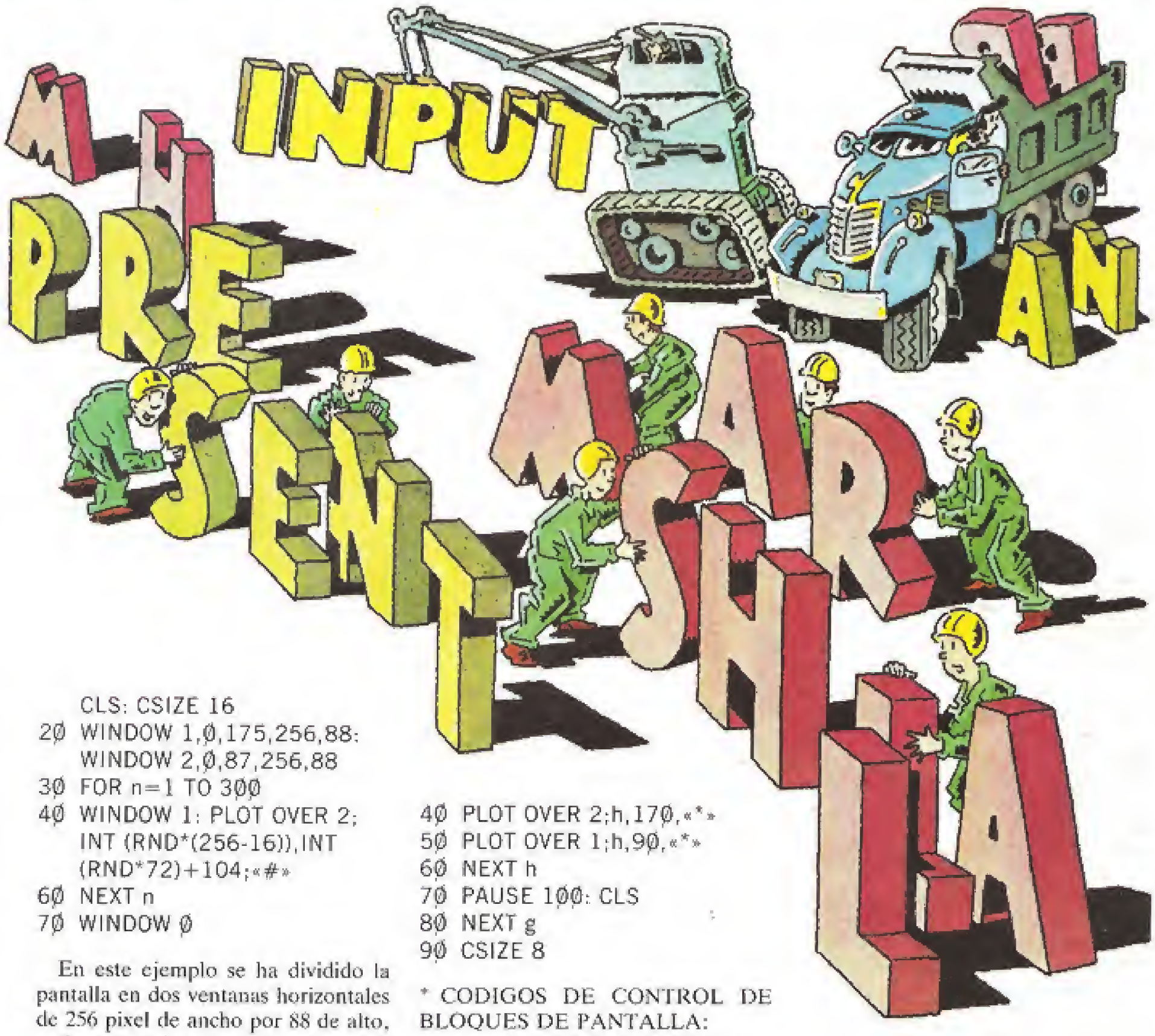
Además podemos especificar mediante t=0 o 1 si queremos que el bloque adquiera los colores en curso de la pantalla (t=0), o si deseamos que tome los atributos del lugar de donde fue obtenido (t=1). Una vez almacenado el gráfico en la variable, su manejo resulta mucho más sencillo (PLOT a$). Ej: Podemos almacenar en a$ un carácter tal como «0» de 32 por 32 pixel (CSIZE 32) y luego imprimirlo aleatoriamente por toda la pantalla (11 veces):

```
1Ø PRINT CSIZE 32; PAPER 6;
   «Ø»
2Ø GET a$,Ø,175,4,4;1: REM
   Con el 1 estamos indicando
   t=1
3Ø BORDER Ø: PAPER Ø:
   CLS
4Ø FOR n=Ø TO 1Ø
5Ø LETx=INT(RND*(256-32)),y
   =INT (RND*(176-32)+32)
6Ø PLOT x,y;a$
7Ø NEXT n
```

El gráfico toma los atributos de la pantalla de origen de donde fue sacado (t=1), y tenemos en cuenta que no podemos imprimir más allá de los límites x=256-(8*4), y=176-(8*4)+32 (podíamos haber utilizado las rutinas de protección de errores mencionadas en el comentario de la versión 1.8)

* OVER 2:
Mezcla gráficos o caracteres con los ya residentes en pantalla realizando una especie de función lógica OR, es decir, si superponemos un pixel sobre otro, el punto seguirá en pantalla, no como con OVER 1 (función lógica XOR) que ante la superposición de dos puntos produce un blanco o inverso.
Ej:

```
1Ø BORDER 7: PAPER 7: INK Ø:
```




```
    CLS: CSIZE 16
2Ø  WINDOW 1,Ø,175,256,88:
    WINDOW 2,Ø,87,256,88
3Ø  FOR n=1 TO 3ØØ
4Ø  WINDOW 1: PLOT OVER 2;
    INT (RND*(256-16)),INT
    (RND*72)+104;«#»
6Ø  NEXT n
7Ø  WINDOW Ø
```

En este ejemplo se ha dividido la pantalla en dos ventanas horizontales de 256 pixel de ancho por 88 de alto, en las que se imprimen en número de 300 veces el carácter «#», pero en la superior mediante el uso de OVER 2 (la pantalla acabará casi ennegrecida), y en la inferior con OVER 1. La diferencia entre ambas queda patente. Sin embargo para aclarar aún más el uso de este comando podemos incluir un segundo ejemplo en el que se aprecia todavía más el efecto de los dos tipos de OVER:

```
1Ø  CSIZE 32,64
2Ø  FOR g=1 TO 4
3Ø  FOR h=Ø TO (256-32) STEP
    (2^g)
4Ø  PLOT OVER 2;h,17Ø,«*»
5Ø  PLOT OVER 1;h,9Ø,«*»
6Ø  NEXT h
7Ø  PAUSE 1ØØ: CLS
8Ø  NEXT g
9Ø  CSIZE 8
```

* CODIGOS DE CONTROL DE BLOQUES DE PANTALLA:

El comando GET almacena bloques de pantalla en forma de caracteres de 8 bytes precedidos por uno de estos códigos de control. CHR$ Ø indica que a continuación vienen 8 bytes de información de pantalla y CHR$ 1 es el indicador de que a continuación se encuentran un byte con la información de los atributos y 8 bytes con la información gráfica.

Ej: Estando en la escala normal podemos almacenar caracteres en variables de la misma forma que definimos los U.D.G. Más tarde los podremos imprimir en la escala, posición y con los atributos deseados.

```
1Ø  BORDER 1: PAPER 1: CLS:
    CSIZE 8
2Ø  LET a$=CHR$ 1+CHR$
    2Ø7+CHR$ 17Ø+CHR$
    85+CHR$ 17Ø+CHR$
    95+CHR$ 17Ø+CHR$
    85+CHR$ 17Ø+CHR$ 85
3Ø  FOR n=Ø TO 1Ø: FOR j=Ø TO
    3Ø
4Ø  PRINT CSIZE 16;AT n,j;a$
5Ø  NEXT j: NEXT n
```

* CODIGOS DE CONTROL

Los ya comentados códigos de control de la versión 1.8 siguen estando a nuestra disposición pero en dos versiones: unos cuyo efecto queda limitado a las distintas ventanas (WINDOW) que hayamos definido, y otros que aun realizando la misma tarea extienden su campo de acción a toda la pantalla.

Sin limitación de ventanas

CHR$ 2 < ..... cursor a la izq.
CHR$ 3 < ..... cursor a la der.
CHR$ 4 < ..... cursor abajo
CHR$ 5 < ..... cursor arriba

Con limitación de ventanas

cursor a la izq. .....> CHR$ 8
cursor a la der. .....> CHR$ 9
cursor abajo .....> CHR$ 10
cursor arriba .....> CHR$ 11

* DRAW TO x,y (,a):

Dibuja una recta (o arco si se especifica «a») llegando al punto «x», «y» prefijado, consiguiendo así un mejor control de la recta a trazar ya que conocemos el punto donde acaba, no como con el DRAW normal con el que elegíamos la longitud en desplazamientos según «x» e «y» a partir de un punto dado, ignorando muchas veces si el pixel de destino se salía o no de la pantalla. Las siguientes rectas que tracemos siguen empezando en el pixel final de la anterior.
Ej: El manual nos ilustra con un buen ejemplo:

```
1Ø FOR N=1 TO 5Ø
2Ø DRAW TO RND*255,
   RND*175: REM Así las rectas
   no se salen
3Ø NEXT N:   REM nunca de la
             pantalla.
```

PROCEDIMIENTOS

* La tendencia actual de muchos ordenadores es la de incluir en su BASIC estructuras propias del PASCAL. En este punto el *Beta Basic* 3.0 introduce grandes mejoras respecto a la versión anterior. Una vez hecha la definición del procedimiento, cada vez que tecleamos su nombre (desde KEYWORDS 3 o 4) éste se ejecutará sin necesidad de introducir el comando PROC (aunque sigue existiendo para mantener la compatibilidad). Si dentro del comando DEF PROC, y a continuación del nombre del procedimiento introducimos otro segundo, éste será tomado como una variable y pasará a formar parte de los llamados «parámetros formales».

Así cada vez que invoquemos un procedimiento deberemos escribir el nombre y un «parámetro actual» que consistirá en un valor numérico o una expresión que dará el valor requerido por el parámetro formal.

Los parámetros incluidos dentro de la definición de un procedimiento son hechos automáticamente variables «locales» que sólo existen dentro de él, por lo que si nuestro programa utiliza el mismo nombre para una variable «global» que el de un «parámetro formal» del procedimiento, el *Beta Basic* se encargará de conservar el valor de la variable «global» y restaurarlo una vez que el procedimiento haya terminado su ejecución. De esta manera se consigue una independencia total de los procedimientos respecto al programa (los dos pueden utilizar las mismas variables sin problema alguno).

Por medio del comando LOCAL podemos crear variables de existencia restringida al ámbito del procedimiento.

Si prevemos que alguna vez uno de los parámetros formales puede quedarse sin valor (parámetro actual), podemos utilizar la instrucción DEFAULT. Actúa como LET pero sólo en caso de que falte el valor asignado a la variable.
Ej:

```
1Ø LET m=2Ø
2Ø DEFAULT k=1Ø, m=15
3Ø PRINT m;« »;k
```

El resultado obtenido será 20 10 ya que m=20 pero «k» al no tener otro valor toma el asignado por defecto en DEFAULT k=10.

Otra posibilidad que nos ofrece el tratamiento de los procedimientos con el *Beta Basic* es la de poder pasar parámetros por referencia, es decir, conseguir pasar algún resultado obtenido en el procedimiento a una variable global del programa. Aunque esto va en contra de la independencia de los procedimientos respecto a las demás rutinas, es una opción muy útil que podríamos llegar a echar en falta. El citado efecto se consigue utilizando el comando REF.
Ej:

```
1Ø DEF PROC calc REF x, REF h
2Ø LET suma=x+h, resta=x-h
25 LET x=suma, h=resta
3Ø END PROC
```

Si una vez definido el anterior procedimiento añadimos las siguientes líneas:

```
1ØØ LET s=25, r=15
11Ø calc s,r
12Ø PRINT s;« »;r
```

Una vez rodado el programa obtendremos el resultado 40 10. Las variables «x» y «h» toman sus valores de las «s» y «r» indicadas en la línea 110, y una vez ejecutado el procedimiento el resultado de nuevo es pasado por «referencia» a «s» y «r».

El comando REF también permite usar matrices de la misma forma, como en REF a( ).

* Procedimientos recursivos:

Constituyen uno de los más interesantes puntos en la utilización del lenguaje PASCAL. De una manera rápida y concisa, la recursión nos permite resolver los cuatro problemas «de llamada» y brillantemente utilizando unas simples líneas de programa. Un procedimiento es recursivo cuando se «llama» o ejecuta a sí mismo hasta o mientras se cumpla una determinada condición o condiciones. Quizá todas las facilidades que proporciona esta estructura para la programación en PASCAL pierdan un poco de sentido en BASIC, pero el *Beta Basic 3.0* no podía dejar de incluirlas.

A continuación añadimos un ejemplo, aunque se trata de un tema sobre

el que habría que hablar mucho más.

```
1Ø  DEF PROC rectangulo x,y
2Ø  DEFAULT x=Ø,y=Ø
3Ø  PLOT x,y
4Ø  DRAW 255-(2*x),Ø
5Ø  DRAW Ø,175-(2*y)
6Ø  DRAW -(255-(2*x)),Ø
7Ø  DRAW Ø,-(175-(2*y))
8Ø  LET x=x+2, y=y+2
9Ø  IF x <(255/2) AND y <(175/
    2) THEN rectangulo x,y
1ØØ END PROC
```

El procedimiento consiste en dibujar un rectángulo centrado en pantalla, y si se dan las condiciones de la línea 90, dibuja otro interior a él y 2 pixel más pequeño mediante la llamada recursiva al mismo procedimiento, pero con los valores de x e y aumentados en dos unidades. El procedimiento termina de autoejecutarse al dejar de cumplirse la condición de control (en este caso condiciones de control). Gracias a la línea 20 podemos ejecutar el procedimiento con teclear simplemente «rectángulo» (con lo que rellenaremos toda la pantalla) o emplear otras llamadas del tipo «rectángulo 10,20».

* Dejando de lado los procedimientos y para resaltar los múltiples usos de muchos de los nuevos comandos, podemos destacar otra nueva utilidad del comando REF. Si éste no es usado dentro de DEF PROC, el ordenador interpretará que queremos encontrar el lugar de nuestro listado donde se halla la referencia apuntada. Así ante un comando del tipo REF a$ el *Beta Basic* editará la línea donde se encuentra «a$» con el cursor parpadeante justo después de la referencia. Con pulsar ENTER dos veces el programa seguirá buscando la siguiente aparición de «a$» hasta que el listado toque a su fin.

MANEJO DE VARIABLES

* ALTER x TO y:

Aparte de su uso gráfico, esta nueva forma de dicho comando nos permite alterar o introducir la modificación indicada por la referencia «y» (puede ser una variable numérica, alfanumérica, una frase, etc...) en la referencia «x». Por lo tanto, para cambiar el nombre de la variable «d$» por el de «p$» no deberemos ir buscando por todo el listado, sino simplemente ejecutar la instrucción ALTER d$ TO p$.

Trabajando con frases el modo de operar es idéntico (por ejemplo, ALTER «Pon tu nombre» TO «escribe tu apellido»). Utilizando paréntesis modificamos sólo el valor de la variable: ALTER (n) TO 5.

* JOIN x TO y; COPY x TO y:

Utilizados con variables y matrices producen efectos semejantes pero con distintos matices. Mientras que JOIN traslada o mueve una variable dentro de otra, el comando COPY introduce una copia o duplicado.
Ej:

```
1Ø  LET s$=«INPUT»,
    b$=«SINCLAIR»
2Ø  JOIN s$(1 TO 3) TO b$
3Ø  PRINT b$
```

Como se puede comprobar, tanto si utilizamos JOIN como COPY el resultado impreso será SINCLAIRINP pero la variable a$ sólo conservará su valor en el segundo de los casos. Utilizando JOIN la variable a$ dejará de existir.

La utilización de estos comandos con matrices se convierte en una eficaz ayuda, facilitando enormemente el traslado o copia de incluso parte de ellas. Ej.:

```
1Ø  DIM h$(14,18): DIM f$(14,
    18)
2Ø  FOR n=1 TO 14: FOR j=1 TO
    18
3Ø  LET h$ (n,j)=STR$ (j+n)
4Ø  NEXT j: NEXT n
5Ø  JOIN f$ TO h$: REM este
    programa muestra como puede
    ser agrandada la matriz h$
    con 14 filas de 18 espacios o
    columnas en blanco
```

Una manera fácil de comprobar el resultado del programa anterior es utilizando una de las funciones del *Beta Basic*, la función LENGTH (explicada más adelante), que nos permite averiguar el número de filas de la nueva matriz formada por la fusión de las dos. Así una vez antes de ejecutar la línea 50, el resultado tras hacer PRINT LENGTH (1,«h$») será 14 mientras que después de la línea 50 el resultado será 28. Además ahora ya podemos tratar con elementos como h$ (20,7) con tan sólo dimensionar la matriz (DIM h$(14,18)).

Con matrices numéricas la utilización de estos comandos se hace de forma idéntica.

* EDIT ;x: EDIT x$:

Según este nuevo formato, el comando EDIT seguido de un nombre de variable (separado por «;» en el caso de variables numéricas para no confundirlo con EDIT número) nos brinda la posibilidad de editar su contenido para una posible modificación. Su utilización con matrices permite listar el contenido de cada fila con sólo teclear EDIT x$(n) siendo «n» el número de la fila elegida. Además su estructura es parecida a la del INPUT permitiendo la introducción de comandos del tipo: EDIT (nombre?);a$

Ej: Si tomamos como ejemplo la matriz JOIN f$ TO h$ del párrafo anterior, podemos modificar su contenido con sólo probar algo como:

```
1Ø  FOR n=1 TO 28: EDIT h$(n):
    NEXT n
```

(haciendo GO TO 1Ø no RUN, para no perder la matriz)

Con esto conseguimos listar uno a uno el contenido de cada fila de la matriz h$. Si una vez editado el contenido de una fila pulsamos ENTER no ocurrirá nada, pero podemos perfectamente modificarlo con simplemente cambiar su valor a modo de un INPUT desde el teclado. Si hacemos EDIT h$(1) tendremos en pantalla «23456-7891111111111». En seguida podemos hacer que la fila h$(1) pase a ser «2345678922222222222» y su valor quedará modificado (a partir de ahora h$(1,18) será 2 y no 1 como antes). También es válido ejecutar instrucciones del tipo EDIT h$(3,9). En ambos

casos deberemos tener en cuenta que la modificación introducida mediante EDIT, quedará limitada por el campo de aquello que estamos editando (introducir 25 valores para h$(1) es aceptado, pero sólo se tienen en cuenta los 18 primeros, los demás se ignoran).

* LET x=5, j=21, g$=«Pedro»

Ahora admite esta estructura que supone un gran ahorro de memoria.

* DELETE:

Además de eliminar líneas de listado, también sirve para borrar variables y matrices o incluso parte de ellas.

Ej:

```
DELETE a$: borra toda la variable
o matriz
DELETE s$ (3): Si s$ es una
variable borrara el tercer caracter,
si es una matriz la tercera
columna
DELETE d$ (2 TO 5): elimina esa
serie de caracteres o esa
«porción» de columnas.
```

* READ LINE:

Con este comando podemos conseguir un gran ahorro de memoria ya que permite introducir una serie de datos alfanuméricos en una sentencia DATA (siempre que empiecen por una letra) sin necesidad de poner comillas. Más tarde el comando READ LINE leerá los datos sin dificultad.

OTRAS FACILIDADES

* El comando DEFAULT puede ser usado como medio para facilitar las operaciones de Microdrives, RS-232 y Red local. Como ejemplo solamente citar que una vez ejecutado DEFAULT=m2 cada vez que hagamos SAVE «programa», en realidad estaremos ejecutando SAVE *«m»;2;«programa» en nuestro microdrive.

NOTA: Así como en el Microdrive funciona perfectamente, los usuarios de la unidad de disco OPUS DISCOVERY pueden encontrar algún problema. El *Beta Basic* 3.0 puede ser grabado en disco con las dos líneas Basic que tiene el programa para tal efecto. Su carga desde disco no presenta ninguna dificultad, pero el resto de opciones presenta algunas deficiencias:

— El comando DEFAULT no funciona con la OPUS (lo cual no es tan grave), aunque por supuesto si lo hace al ser usado en los procedimientos.

— Para hacer un catálogo del disco deberemos teclear CAT «m»;1 en vez del normal CAT 1.

— Si intentamos grabar en un disco que está protegido obtendremos un mensaje ilegible y no sabremos qué nos está ocurriendo. Procura estar atento si no quieres «volverte loco» cuanto te pase.

Por lo demás, el *Beta Basic 3.0* es una maravillosa herramienta de trabajo, sobre todo cuando la usamos con el disco. De todas maneras, si tuvieras alguna vez problemas al intentar grabar un fichero desde el programa, sabes que siempre puedes desconectarlo (RANDOMIZE USR 59904), grabar tu fichero y volver a activar el programa (RANDOMIZE USR 58419).

* CLEAR x:

Si dicho comando va seguido por un dígito de tres cifras bajará el RAMTOP el número de bytes indicado. Si el número es negativo subirá el RAMTOP la cantidad de posiciones indicada.

* ON x:

Se ha introducido una nueva versión de este comando aparte de lo ya apuntado en el *Beta Basic 1.8*. Según el valor que tome la variable «x» el comando ON dirigirá el control del programa al lugar indicado dentro de la multisentencia que se encuentre a continuación.

Ej:

```
1Ø INPUT x
2Ø ON x: PRINT «SINCLAIR»: GO
   TO 1Ø:
3Ø STOP
```

Si x=1 imprimirá SINCLAIR, si x=2 irá a la línea 10.

* LIST / LLIST: Sus funciones han sido aumentadas en gran número, aunque dentro del mismo campo:

— LIST / LLIST PROC nombre: Lista las líneas que componen un procedimiento determinado.

— LIST / LLIST DEF KEY: Lista los contenidos de las teclas predefinidas con el comando DEF KEY.

— LIST / LLIST REF: Genera una lista de números indicando las líneas donde aparece la referencia apuntada. Si la referencia aparece varias veces en una misma línea, entonces el número se repite.

— LIST / LLIST DATA: Proporciona información sobre los distintos tipos de variables y matrices:

1. Confecciona una lista con los nombres y dimensiones de las matrices numéricas.

2. La lista sigue con un estudio detallado de las variables de control de los bucles FOR-NEXT, indicando el nombre, valor de STEP y valor inicial, además del número de línea a la que salta cada vez que se realiza el bucle.

3. A continuación añade los nombres y contenidos de las variables numéricas.

4. Continúa la lista con una sección alfabética de las variables numéricas cuyo nombre tiene más de un carácter, además de indicar su contenido.

5. Especifica el nombre y dimensión de las matrices alfanuméricas.

6. Termina el listado con una sección que destaca el nombre, longitud y contenido de las variables alfanuméricas.

— LIST / LLIST VAL: Ejecuta la misma función que el comando anterior, pero restringido sólo a las variables y matrices numéricas (es decir, los 4 primeros puntos).

—LIST/LLIST VAL$: Como el anterior, pero respecto a las alfanuméricas (puntos 5 y 6).

— LIST / LLIST FORMAT x: Como su traducción indica, según el valor de «x» produce distintos tipos de listado:

FORMAT 0: Produce un listado en el que los números ocupan las 5 primeras columnas y el resto de las sentencias empiezan a partir de la sexta, aunque ocupe varias líneas.

FORMAT 1: Representa cada parte de una multisentencia en una línea separada, es decir, actúa como una especie de ENTER cuando encuentra «:». Con esto sin duda se consigue una mayor visión global.

FORMAT 2: Así como en el modo de formato anterior, hay una serie de comandos tales como FOR, DO o DEF PROC que causan el que las sentencias correspondientes a su «rango», aparte de situarse en otra línea, se vean desplazadas una columna respecto a las demás, cosa que ocurre hasta que se termina el «rango» (hallamos a LOOP, END PROC, etc.).

En el caso de FORMAT 1 el desplazamiento es el indicado. Con FORMAT 2 las sentencias se trasladan 2 columnas. En ambos casos podemos decir que se ha producido una indentación, típica de programas escritos en lenguajes como el PASCAL.

FORMAT 3, 4 y 5: Actúan como los casos 0, 1 y 2 pero sin mostrar los números de línea.

```
FORMAT Ø
  5Ø PRINT «El Beta Basic 3.Ø es
     un programa ampliador del
     Basic de tu ordenador»
  1Ø DEF PROC prueba
  2Ø FOR n=Ø TO 1Ø: PRINT
     «INPUT»: PRINT «SINCLAIR»:
     NEXT n
  3Ø END PROC

FORMAT 1
  1Ø DEF PROC prueba
  2Ø FOR n=Ø TO 1Ø
     PRINT «INPUT»
     PRINT «SINCLAIR»
     NEXT n
  3Ø END PROC

FORMAT 2
  1Ø DEF PROC prueba
  2Ø FOR n=Ø TO 1Ø
     PRINT «INPUT»
     PRINT «SINCLAIR»
     NEXT n
  3Ø END PROC
```

CINCO NUEVAS FUNCIONES

La última versión del *Beta Basic* introduce 5 nuevas funciones sin modificar las ya existentes:

* EOF (x): Función que indica el «End of file» o final del fichero del Discovery o Microdrive que estamos explorando. «x» ha de ser el número del canal y la función genera un «1» si el último dato del fichero ha sido leído, o un «0» si no fue así. Utilizando este comando podemos conseguir evitar ciertos errores en la lectura de ficheros, como el producido al intentar leer más registros de los exis-

tentes. Por ejemplo, una vez abierto un hipotético canal OPEN #7,«m»;1; «teléfonos» para leer dentro del fichero, si queremos prevenir errores podemos emplear instrucciones del tipo:

IF EOF(7)=0 o 1 THEN GOTO, DO UNTIL EOF(7)=1 o incluso DO WHILE EOF(7)=0

* INARRAY: Actúa de manera semejante a INSTRING con la particularidad de hacerlo dentro de matrices. La serie de caracteres a buscar puede estar representada con comodines («wild cards»), es decir, el carácter «#» sustituyendo algún elemento de la «string». Cuando dicha serie de caracteres es encontrada dentro de la matriz, el número indicando la fila donde se halla puede ser almacenado en una variable para su posterior manipulación.

Si la «string» no es encontrada la función devolverá un 0.

* ITEM( ): Es una especie de EOF para ser utilizado dentro de sentencias DATA. Cuando todos los datos han sido leídos, la función devuelve un 0. Si no es así genera un 1 cuando el siguiente dato a leer es de tipo alfanumérico o un 2 si es numérico.

* LENGTH(x,«nombre»): Informa acerca de las dimensiones de una matriz determinada («nombre»). Si x=1 estamos requiriendo datos sobre la primera dimensión (filas) y si x=2 sobre la segunda (columnas).

```
1Ø DIM a$ (3Ø,1Ø)
2Ø PRINT LENGTH(1;«a$»);«  »;
   LENGTH(2;«a$»)
3Ø REM imprimirá 3Ø  1Ø
```

* SHIFT$(x,a$): Es una muy útil función que nos permite tener un control más directo sobre cualquier tipo de variable alfanumérica. Según el valor de «x» se consiguen distintos resultados, como pasar de mayúsculas a minúsculas (o viceversa), evitar los errores derivados de la impresión de un carácter desconocido (pues puede ser un carácter irrepresentable que da el mensaje «Invalid colour»), transformar los «tokens» en su forma deletreada (para evitar posibles errores de impresión) o al contrario, para pasar a la forma «tokenizada», etc...

Quizá el uso más interesante sea el apuntado en último lugar, que nos permite pasar a la forma tokenizada y viceversa evitándonos así todos los problemas derivados del uso de impresoras distintas de la ZX-Printer o GP-50 y la impresión de los nuevos comandos incluidos por el *Beta Basic*.

El proceso consiste en abrir un fichero en el disco o microdrive y listar en él nuestro programa a imprimir:

```
OPEN #5; «m»;1;«listado»: LIST
#5;: CLOSE #5
```

Una vez hecho esto y para no tener problemas (sobre todo con la OPUS) haremos CLEAR #, y más tarde teclearemos (o cargaremos) el programa listador, el cual al ser ejecutado producirá el listado en nuestra impresora:

```
1Ø OPEN # 3;«t»: REM abrimos
   el canal «t» que ejecuta un
   CHR$ 13
2Ø OPEN #5;«m»;1;«listado»
3Ø LET a$=«  »
4Ø DO UNTIL EOF(5)=1
5Ø INPUT #5; LINE a$
6Ø LPRINT SHIFT$ (7,a$)
7Ø LOOP
8Ø CLOSE #5
```

Esperamos que con este artículo y el ya publicado sobre el *Beta Basic 1.8* hayamos podido introducir más al lector en el manejo de uno de los mejores programas disponibles para este ordenador.

GANADORES DE LOS MEJORES DE INPUT SINCLAIR

En el sorteo correspondiente al número 19 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Vidal Jerez Cid	Salamanca	TRIVIAL PURSUIT
Jesús Rodríguez Herrero	Basauri (Vizcaya)	TRIVIAL PURSUIT
Gabriel José Castejón López	Sevilla	FERNANDO MARTIN
Alfonso Manuel Delgado Lozano	Sevilla	EL MISTERIO DEL NILO
Antonio Casado Martínez	Guadix (Granada)	MEGA BUCKS
Raúl Villarreal Martín	V. de la Sagra (Toledo)	FERNANDO MARTIN
Ramón Bravo Bautista	Madrid	STRIKE FORCE HARRIER
Jorge Saavedra Díaz	Barcelona	GHOSTS'N GOBLINS
José Antonio Gordillo López	Sevilla	THE GREAT ESCAPE
Ferrán García Miracle	Vilanova i La Geltrú (Barcelona)	FIST II

chip

¡DEBERÍA DARTE VERGÜENZA!

SI NO TIENE...

¡COMO SE ENTERE EL CPU TE VAN A CAER LAS PATILLAS...!

¡UY, SI SE ENTERA...!

¡¿PERO TÚ QUE TE CREES QUE ES ESTO?!

¡¿LA FERIA DE SEVILLA?!

SÓLO LE FALTAN LAS CASTAÑUELAS

YA OS HE DICHO MIL VECES QUE YO NO TENGO LA CULPA...

¡BAH! NO NOS VENGAS OTRA VEZ CON LA TONTERÍA ESA DEL SIMULADOR DE VUELO...

¡QUE SÍ, HOMBRE! LO QUE PASA ES QUE A VOSOTROS APENAS OS HA AFECTADO... YA VERÉIS LA PRÓXIMA VEZ QUE...

¡ESCUCHAD! PARECE QUE VA A JUGAR DE NUEVO...

¡AJÁ! YA HA EMPEZADO... ¡EL AVIÓN ESTÁ DESPEGANDO...!

BRRRRRRRRRR

RRRRRRRRRRRR

AHORA SE VA ACERCANDO A UN ARCO IRIS...

... Y DENTRO DE POCO LO ATRAVESERÁ...

¡AHORA! ¿NO LO NOTÁIS EN LOS CIRCUITOS?

S... S... SÍ...

¿QUÉ? ¿OS CONVENCÉIS AHORA? YA OS DIJE QUE ESE ARCO IRIS DESTEÑÍA...

¿Y... Y ESO NO SE VA?

OYE... PUES NO QUEDA MAL...

¡JO!

SIRVENT '87

RUTINAS EN MEMORIA ROM (II)

Con esta segunda parte continuamos el artículo iniciado en el pasado número donde iniciamos el estudio de las rutinas de la ROM del Spectrum. **Seguimos repasando las rutinas generales de ejecución.**

4535-4554 ($1167-$11CA).- Rutina de tratamiento de la instrucción NEW.

4555-4559 ($11CB-$11CF).- Rutina que coloca el BORDE de color blanco, START/NEW.

4560-4569 ($11D0-$11D9).- Rutina de inicialización del registro indexado I, se inicializa con ($3F), se usa en la generación de señal de la TV.(espera de 4t estados).

4570-4591 ($11DA-$11EF).- Rutina de cálculo de la memoria disponible y el resultado lo coloca en las variables del sistema, es decir mueve la RAM-TOP, RAM-CHECK.

4592-4607 ($11F0-$11FF).- Rutina de restauración de valores iniciales de P-RAMT, RASP, PIP y UGD, con la instrucción NEW.

4608-4632 ($1200-$1218).- Rutina de restauración y asignación de gráficos usuario, a las letras, (A-U).

4633-4660 ($1219-$1234).- Rutina para inicializar la variables CHARS ($3C00) y controla las interrupciones cada 1/50 de segundo.

4661-4675 ($1235-$1248).- Rutina para copiar el canal de información en memoria.

4676-4731 ($1244-$1278).- Rutina que restaura en las variables del sistema el valor inicial. (INK 7: PAPER 0: BORDER 0).

4732-4741 ($127C-$1285).- Rutina que transfiere datos a las 14 primeras posiciones, indicadas por la variable STRMS.

4742-4769 ($1286-$12A1).- Rutina que restaura el buffer de impresora, la pantalla y saca el mensaje de inicialización.

4770-4833 ($12A2-$12E1).- Rutina que controla y lista las secuencias del BASIC para que sean correctas las sentencias MAIN-EXEC.

4834-4866 ($12E2-$1302).- Rutina que chequea y testea, con el INTERPRETE, las líneas BASIC, saltando a la rutina $1303 en caso de error.

4867-5008 ($1303-$1390).- Rutina que reporta y señaliza el error para su correción, retorna a la rutina 12AC$.

5009-5468 ($1391-$155C).- Tabla de mensajes de ERROR, incluyendo el copyright de iniciación de pantalla con un total de 31 mensajes.

5469-5550 ($155D-$15AE).- Rutina que almacena un línea nueva de BASIC a la memoria MAIN-ADD.

5551-5573 ($15AF-$15C5).- Rutina de inicialización de los canales K,S,R,P. Es decir el TECLADO, la PANTALLA, espacio MEMORIA, y la IMPRESORA.

5574-5587 ($15C6-$15D3).- Tabla de datos con los contenidos de las variables para la inicialización.

5588-5605 ($15D4-$15E5).- Rutina de control de la línea editada por la instrucción INPUT, WAIT-KEY.

5606-5614 ($15E6-$15EE).- Rutina para salvar los registros, colocando en el registro HL la dirección de la zona de edición, almacenada en la variable del sistema (CURCHL). INPUT-AD.

5615-5632 ($15EF-$1600).- Rutina del tratamiento y almacenaje del registro A, en operaciones de ENTRADA/SALIDA.

5633-5652 ($1601-$1614).- Rutina de apertura del canal de salida de datos (1,2,3) , contenido en el acumulador, CHAN-OPEN.

5653-5676 ($1615-$162C).- Rutina de inicialización de los FLAGS, de los canales en la variable del sistema FLAGS2.

5677-5683 ($1620-$1633).- Lista de las direcciones de las rutinas de inicialización de los canales.

5684-5697 ($1634-$1641).- Rutina de inicialización del teclado, canal K.

5698-5708 ($1642-$164C).- Rutina de inicialización de la pantalla, canal 5.

5709-5713 ($164D-$1651).- Rutina de inicialización de la impresora, canal P.

5714-5731 ($1652-$1663).- Rutina de inicialización del espacio de memoria, canal R, ONE-SPACE.

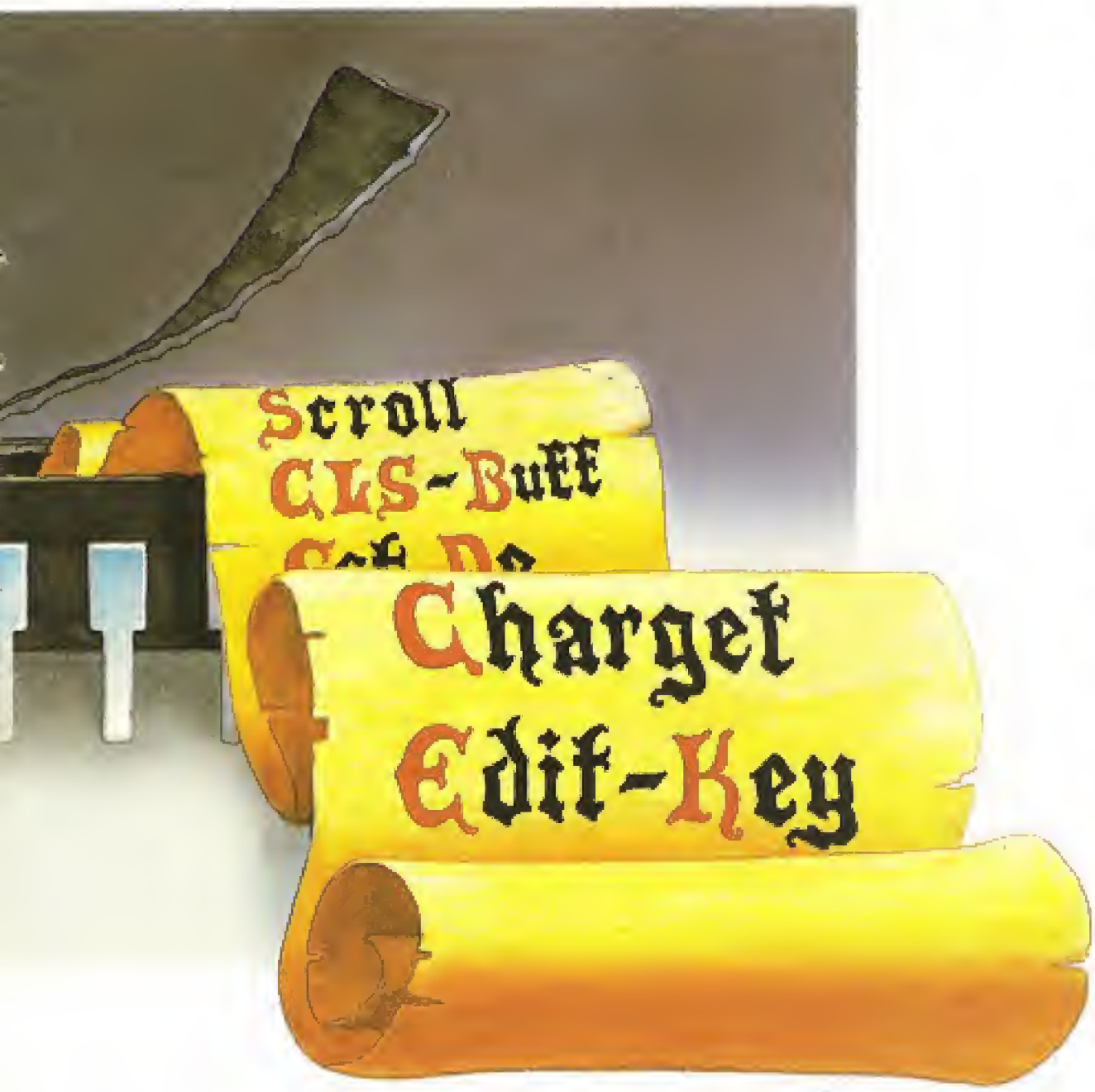
5732-5774 ($1664-$168E).- Rutina de modificación del emplazamiento de los punteros, POINTERS.

5775-5789 ($168F-$1690).- Rutina que almacena en el registro DE, el número de línea apuntada por el registro HL.

5790-5807 ($169E-$16AF).- Rutina que reserva espacio de memoria, llamada por la rutina RST0030, RESERVE.

5808-5850 ($1680-$16DA).- Rutina de inicialización de las variables que apuntan a la zona de edición, SET-MIN.

5851-5860 ($16D8-$16E4).- Rutina de exploración de la tabla apuntada por el registro HL, INDEXER.



5861-5888 ($16E5-$1700).- Rutina del comando CLOSE.

5889-5909 ($1701-$1715).- Rutina del cierre de los canales especificados. CLOSE-2.

5910-5917 ($1716-$1710).- Lista de las direcciones de las rutinas de cierre de los diferentes canales, CLOSE.

5918-5941 ($171E-$1735).- Rutina que almacena en el registro BC, el dato del canal especificado por el registro A, STREAM-DATA.

5942-6009 ($1736-$1779).- Rutina del comando OPEN.

6010-6034 ($177A-$1792).- Tabla de las direcciones de las rutinas de apertura de los canales. OPEN.

6035-6036 ($1793-$1794).- Salto a las rutinas de las instrucciones CAT, ERASE, FORMAT, MODE.

6037-6132 ($1795-$17FA).- Rutinas del tratamiento de las instrucciones LIST, LLIST.

6133-6136 ($17F5-$17F8).- Punto de entrada a la rutina por la instrucción LLIST.

6137-6228 ($17F9-$1854).- Punto de entrada a la rutina por las instrucción LIST.

6229-6325 ($1855-$1865).- Rutina de presentación en el canal abierto de una línea BASIC, OUT LINE.

6326-6336 ($1886-$18C0).- Rutina de gestión y tratamiento de los números en coma flotante en una línea BASIC, NUMBER.

6337-6368 ($18C1-$18E0).- Rutina de presentación en la pantalla de un caracter parpadeante.

6369-6414 ($18E1-$19DE).- Rutina de presentación en pantalla del cursor.

6415-6436 ($190F-$1924).- Rutina de modificación del contenido de las variables del sistema (S-TOP, E-PPC), almacenando el número de línea siguiente, el registro HL apunta a las variables.

6437-6509 ($1925-$197F).- Rutina de presentación en pantalla de los caracteres de una línea BASIC.

6510-6527 ($196E-$197F).- Rutina de búsqueda en memoria de una línea BASIC, el número de línea en registro HL, LINE-ADDR.

6528-6535 ($1980-$1987).- Rutina de comparación en la búsqueda de una línea en memoria, de los octetos en la memoria y de la línea.

TU ERES FANTASTICO

¡Unete a LOS 4 FANTASTICOS!

CADA MES EN TU QUIOSCO

COMICS forum

EL JUEGO DEL ZORRO Y LAS OCAS (I)

- EXPLICANDO EL JUEGO
- EVALUANDO POSICIONES DE JUEGO
- ARBOL DE INVESTIGACION
- EJECUTANDO EL PROGRAMA

Tu ordenador, para jugar, puede convertirse en un «zorro» astuto o en temerosas ocas.

Observa cómo se pueden aplicar métodos de Inteligencia Artificial a programas escritos capaces de pensar.

En un número anterior de INPUT viste cómo era posible escribir un programa Otelo, en el que el ordenador jugaba de un modo bastante aceptable contra ti. En cualquier caso, con un poco de práctica, seguro que fuiste capaz de vencer al ordenador la mayoría de las veces.

En el curso de las tres partes de programación de JUEGOS que forman este juego verás cómo también se puede escribir un programa más sofisticado, a fin de que tu ordenador pueda jugar contra ti el Juego del Zorro y las Ocas. El programa tiene muchos niveles de habilidad, de modo que puedes hacer que el juego sea tan difícil o tan fácil como quieras.

El juego del Zorro y las Ocas ilustra muchos de los problemas y principios que se incluyen a la hora de escribir uno de los más interesantes y perdurables programas de juegos: el ajedrez. La adquisición de un programa de ajedrez comercial de tipo medio, ejecutable en un microordenador, vale la pena, a no ser que los jugadores posean un nivel muy bueno en ajedrez, o conozcan bien las debilidades de los programas de ajedrez.

No se puede escribir un programa de ajedrez que valga la pena en BASIC, pues el ordenador va a tardar un siglo en realizar cada movimiento. En los niveles superiores del Zorro y las Ocas observarás que el programa tarda mucho más para decidir cada uno de sus movimientos, tal vez más de media hora en los niveles más superiores ejecutados en ordenadores más lentos. El programa aún sería mucho más lento si, en lugar de este juego, intentases escribir un programa de ajedrez para ordenador.

Necesitamos un juego más simple para mantener alejado nuestro juego-ejercicio del realismo del código máquina. El Zorro y las Ocas nos viene de perlas porque posee muchas de las características del ajedrez, e incluso se juega con el mismo tipo de tablero.

Este artículo se divide en tres partes.

En esta primera estudiarás los principios que se encierran en un programa de este tipo y que se irán desarrollando en la segunda y tercera parte del artículo, hasta configurar un juego completo. Pero comenzamos observando el juego en sí mismo y todo lo que requiere el programa para que se pueda jugar con él.

EL JUEGO

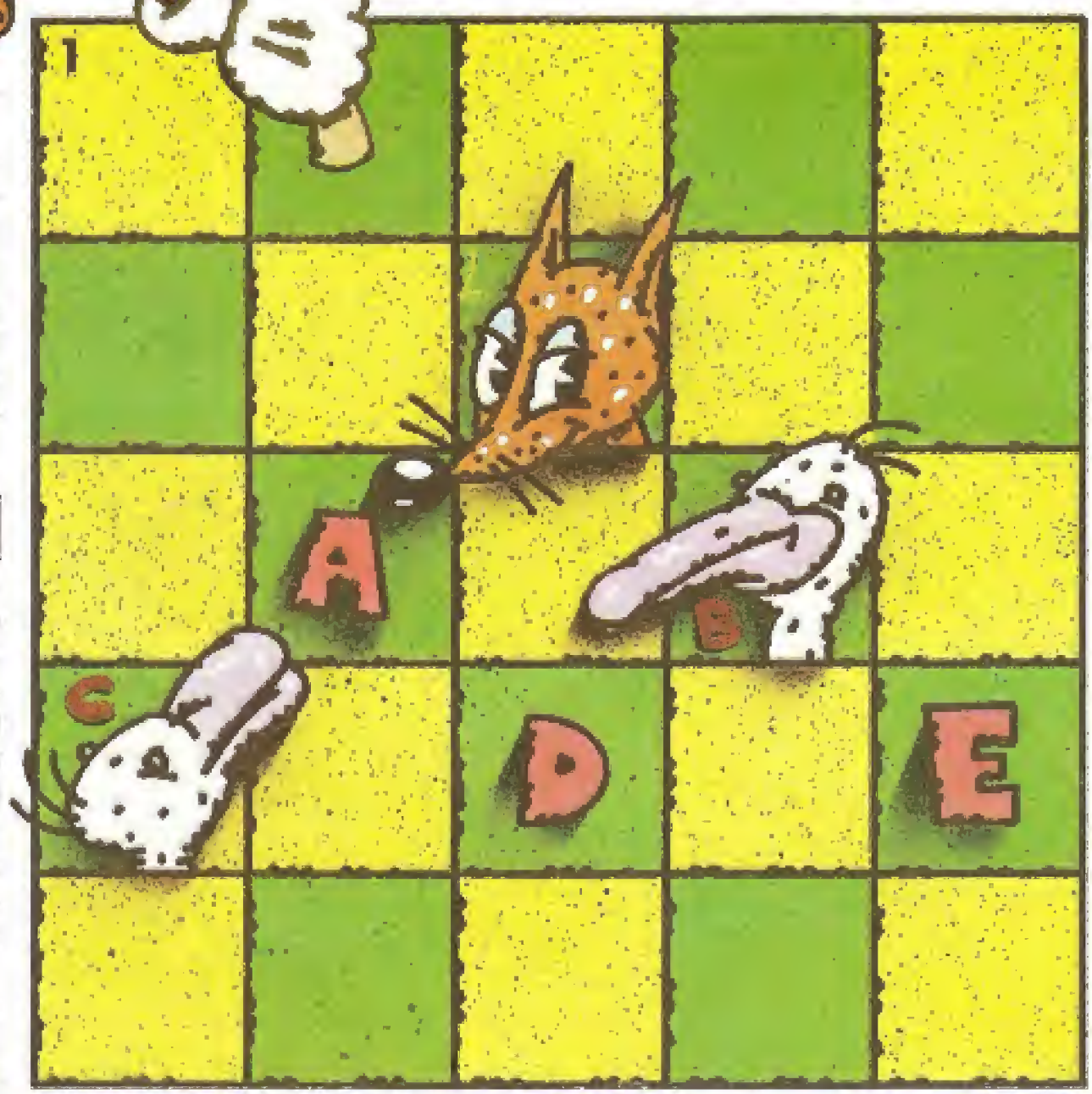
El juego del Zorro y las Ocas se juega sobre los cuadros blancos de un tablero de ajedrez. Hay un Zorro que empieza desde una esquina del tablero y cuatro Ocas que comienzan desde el lado opuesto. Un jugador controla al Zorro y el otro a las cuatro Ocas. El objetivo del juego consiste en lo siguiente: el Zorro ha de pasar entre las Ocas para llegar al otro lado del tablero, y la misión de las Ocas consiste en rodear al Zorro.

Con cuatro contra uno, el juego puede parecer un poco injusto, pero resulta que los movimientos de las Ocas están limitados a avanzar únicamente hacia delante, mientras que el Zorro se puede mover tanto hacia delante como hacia atrás. El programa está escrito de tal modo que el ordenador puede tomar el puesto, tanto del Zorro como de las Ocas, e incluso puede programarse para jugar contra sí mismo.

ABORDANDO LOS PROBLEMAS

El juego del Zorro y las Ocas es un pequeño ajedrez en el sentido de que no existe elemento de suerte en el juego: el resultado depende totalmente de la habilidad de los jugadores. Aunque en teoría, el ordenador podría aprender a jugar a base de probar y equivocarse, como un jugador humano, en el estado actual, éste no es el mejor camino para resolver los problemas que plantea este juego, y además, ¡el programa no cabría en tu microordenador...!

Programar un juego como el del Zorro y las Ocas, o el del ajedrez, es realmente un ejercicio de Inteligencia Artificial. Para ofrecer un firme desafío a un oponente humano, el ordenador deberá ser capaz de jugar *inteligentemente*. Desgraciadamente la máquina no puede mirar al tablero y absorber una relación de espacio entre las piezas, como tú haces. En su lugar,



la posición en el tablero se convierte en valores numéricos que el ordenador puede entender.

Si quieres escribir un programa para que el ordenador juegue «inteligentemente», deberás tener en cuenta primero la naturaleza del juego. El tipo de juego dictaminará el tipo de programa.

Tal vez exista un *método mejor*, totalmente documentado, que puedas adaptar a tu ordenador. Otros candidatos similares a este tipo de tratamiento serían cosas como: resolver un cubo Rubik, jugar al tres en raya, o las aperturas en ajedrez. En tales casos, tu trabajo se verá bastante simplificado. Si existe un gran elemento de suerte en el juego, es posible utilizar la estrategia de *un solo movimiento*, en la que el programa examina todas las posibilidades abiertas que diseñase para que elija entre el mejor movimiento. Los juegos *Ludo* y *Monopoly* podrían ser candidatos preferibles de *un solo movimiento*. Juegos de este tipo requieren comparativamente rutinas de decisiones simples.

En los juegos en los que entran pocos o ningún elemento de azar, como el del Zorro y las Ocas, deberás utilizar una estrategia del «movimiento consecutivo», observando entre una serie de movimientos, a fin de investigar las posibles salidas. El programa halla el movimiento más ventajoso desde cada posición y lo realiza.

EVALUACION DE LA POSICION

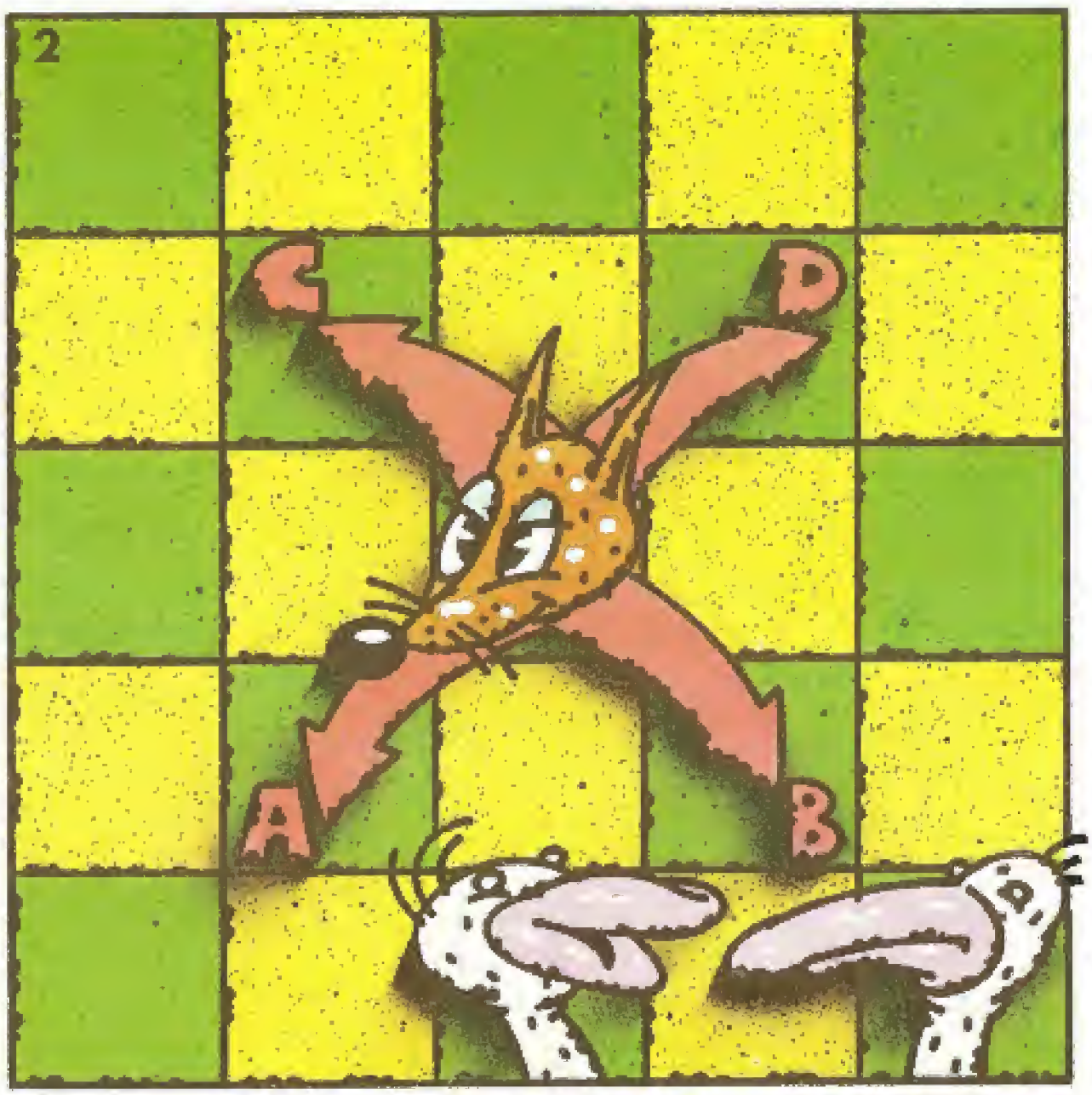
A fin de permitir al ordenador decidir cuál es el mejor movimiento dentro de una categoría de posibilidades, a cada cuadro del tablero se le asigna un valor numérico. En el Zorro y las Ocas, cuanto más abajo se halle el Zorro, tanto mejor para él, pues recuerda que éste gana cuando llega al otro extremo del tablero.

Las filas de casillas están numeradas alternativamente de izquierda a derecha y luego de derecha a izquierda. Ello asegura que las Ocas tiendan a permanecer en una línea recta —su configuración más fuerte— y el cuadro que llevará asignando el valor más alto se va alternando entre el final izquierdo y derecho de la fila.

Además de la sencilla evaluación posicional, los cinco cuadros blancos delante del Zorro tienen un significado especial en el juego. Si observas la figura 1, verás cinco cuadrados con las letras de A a E.

Si sólo hay una Oca situada en estos cuadros, gana el Zorro; si hay dos Ocas, y no están en las posiciones A o B; o bien, si hay tres Ocas y éstas están en las posiciones ACD, BDE, ACE o BCE, el Zorro aún sigue ganando.

Al comienzo de cada turno de juego, cuando el ordenador juegue, o bien a ser Zorro o las Ocas, el programa salta a una subrutina que revisa la posición (o configuración) de todas las piezas y las convierte en un solo número que utiliza el programa para decidir qué movimiento realizar. Una vez tomada la decisión, el número vuelve a convertirse en una configuración.



ARBOL DE INVESTIGACION

Los posibles movimientos de posición sobre el tablero pueden ser representados con una escritura en forma de árbol, cuyas ramas emanan de la posición de las piezas (raíces). Por ejemplo, desde la posición en la figura 2, se podría trazar el primer nivel del árbol en la figura 3. Si miras dos movimientos más adelante el árbol se vuelve más complejo que en el segundo nivel; ten en cuenta que no siempre hay razón para que partan cuatro ramas de cada cuatro, ya que la pieza podría estar bloqueada por otra, o podría hallarse en el extremo del tablero.

ACELERANDO EL PROGRAMA

Con el número de cálculos a ejecutar, el BASIC puede resultar muy lento, pero existen tres caminos para

acelerar el programa. En primer lugar, puedes asegurarte de que el programa no molesta para evaluar el próximo movimiento del adversario, en caso de que tu movimiento haya ganado 1 partida. En cualquier caso, ella sólo ahorra tiempo al final, pero no durante la partida.

En segundo lugar, se puede llegar muy frecuentemente a la misma configuración desde muy diversas vías. Si éste es el caso, valdrá la pena el esfuerzo de construir una tabla de configuraciones comunes y de valores calculados asociados con ellos. Una tabla de este tipo detiene el programa volviendo a asignar cada vez la posición individualmente, pero no hay duda en la conveniencia de grabar una configuración que requiere menos tiempo en ser evaluada que lo que se tardaría en encontrarla en la tabla. En teoría, la tabla sólo puede ser utilizada cuando el programa ha avanzado tres *pliegues* (turnos completos de cada jugador) o más, la práctica del juego demuestra que no hay razón de registrar nada en la tabla hasta que se hayan avanzado, como mínimo, cinco *pliegues*.

En tercer lugar, se puede utilizar el denominado algoritmo alfa-beta. Éste fue descubierto a principios de los 60 por investigadores del campo de Inteligencia Artificial, y se emplea cada vez que el registro del árbol necesita examinar más de un *pliegue* de un árbol.

Si observas la figura 3 podrás ver la evaluación de los posibles movimientos de una posición del Zorro. El programa evaluará completamente la rama A y luego pasará a la rama B. Si en cualquiera de las fases de evaluación de B se detecta un resultado erróneo, se rechazarán todos los movimientos en B. El mejor movimiento efectuado hasta el momento queda registrado en la memoria del programa y comparado con los resultados en cada rama en turno. Si se encuentra en un punto cualquiera un resultado erróneo, ello provocará que toda la rama sea rechazada.

El algoritmo alfa-beta adquiere mayor importancia realmente a medida que aumenta la complejidad del árbol. Si el árbol aumenta considerablemente de tamaño, puede permitirte descartar alrededor del 99,8 % de posibilidades a un nivel primario, con un ahorro similar de tiempo. Pero en este juego el árbol no se aproximará a este nivel de complejidad.

EJECUTANDO EL PROGRAMA

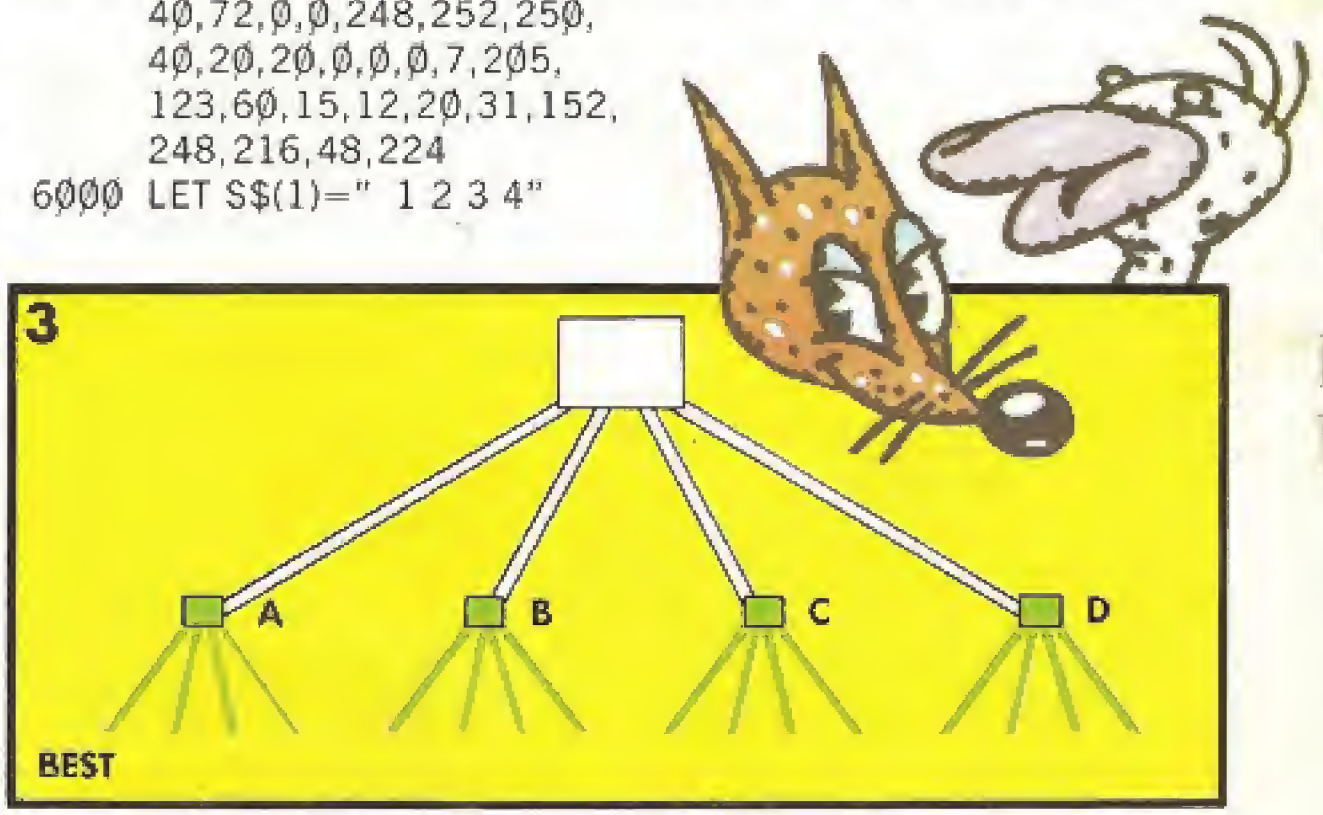
Ahora que ya tienes algo de idea acerca de la teoría que alberga la escritura del programa de un juego de estas características, puedes pasar a la primera sección del programa. Consiste en trazar los gráficos, aunque no verás nada en este nivel. Si ejecutas el programa con RUN, no olvides grabarlo (SAVE) en un cassette que utilizarás en la próxima parte del programa.

```
1Ø  DEF FN A(F)=INT (LN (F)/
    L2+.ØØ1)
1ØØ  GO TO 2ØØ2
2ØØ2  GO SUB 5ØØØ
2ØØ6  BORDER 4: PAPER 4: CLS
2ØØ8  PRINT AT 8,1; FLASH 1;
      "ESPERA UN MOMENTO
      POR FAVOR"
4ØØØ  PRINT AT 21,Ø;"
4Ø1Ø  RETURN
5ØØØ  FOR J=USR "A" TO USR
      "D"+7: READ A: POKE J,
      A: NEXT J: RETURN
5Ø7Ø  DATA 2Ø,28,55,127,15,2Ø
      4Ø,72,Ø,Ø,248,252,25Ø,
      4Ø,2Ø,2Ø,Ø,Ø,Ø,7,2Ø5,
      123,6Ø,15,12,2Ø,31,152,
      248,216,48,224
6ØØØ  LET S$(1)="  1 2 3 4"
6Ø1Ø  LET S$(2)="8 7 6 5"
6Ø15  LET S$(3)"  9 1Ø 11 12"
6Ø3Ø  LET S$(4)="16 15 14 13"
6Ø4Ø  LET S$(5)="  17 18 19
      2Ø"
6Ø5Ø  LET S$(6)="24 23 22 21"
6Ø6Ø  LET S$(7)="  25 26 27 28"
6Ø7Ø  LET S$(8)="32 31 3Ø 29"
61ØØ  RETURN
```

VISUALIZANDO EL TABLERO

```
31Ø  LET F=FN A(ABS (P))-3Ø:
     LET B=P/B(F): IF B<Ø
     THEN LET B=B+ BX: LET
     F=33-F
32Ø  LET C=B/B(29): FOR A=8
     TO 1 STEP -1: LET
     R$(A)=B$(INT (C)+1,
     (2-FN W(A))): LET C=(C-
     INT (C)) *16: NEXT A
33Ø  LET R$(INT (F/4+.8))(FN
     C(F)+1 TO FN
     C(F)+4)=F$(FN W(F/4-
     .2)+1)
34Ø  FOR A=1 TO 8: PRINT AT
     2*A, 8; PAPER 7;S$(A):
     PRINT AT 2*A+1,8; PAPER
     7;R$(A): NEXT A: RETURN
```

Las líneas 310 a 350 muestran el tablero con las cinco piezas en posición. La subrutina es llamada una vez a cada turno del Zorro y las Ocas.

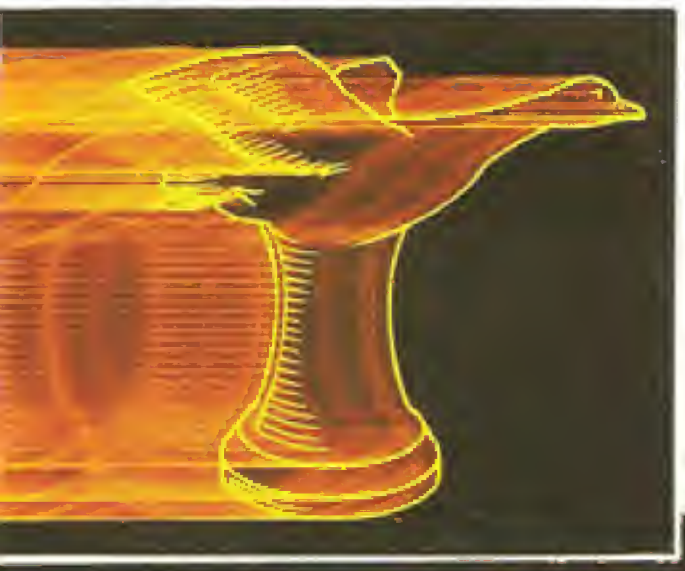


EL ZORRO Y LAS OCAS (II)

Haz uso de la última parte de la teoría para empezar a escribir el juego del Zorro y las Ocas.

He aquí la rutina para inicializar el juego y para trazar Movimientos.

Esta vez vas a introducir rutinas de inicialización y la rutina principal de trazado. Asimismo, también podrás proponer al jugador otra partida, pero en este estadio aún no podrás ejecutar con RUN el programa, pues aún hay varias rutinas importantes que añadir.

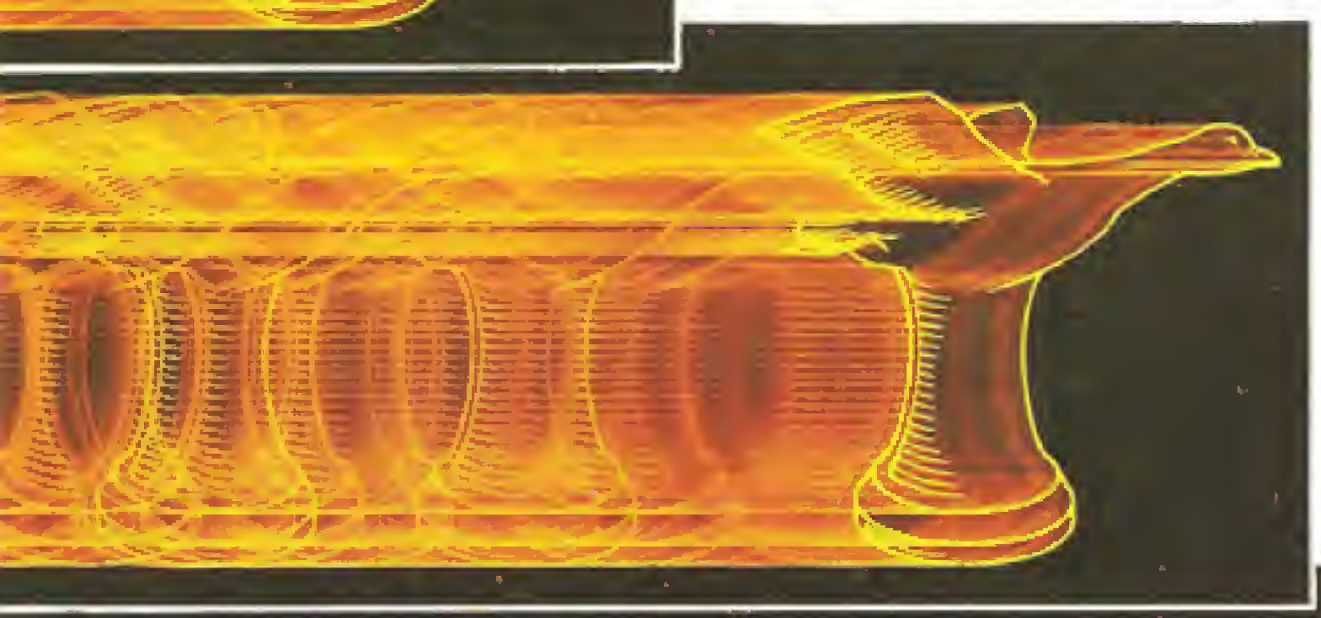


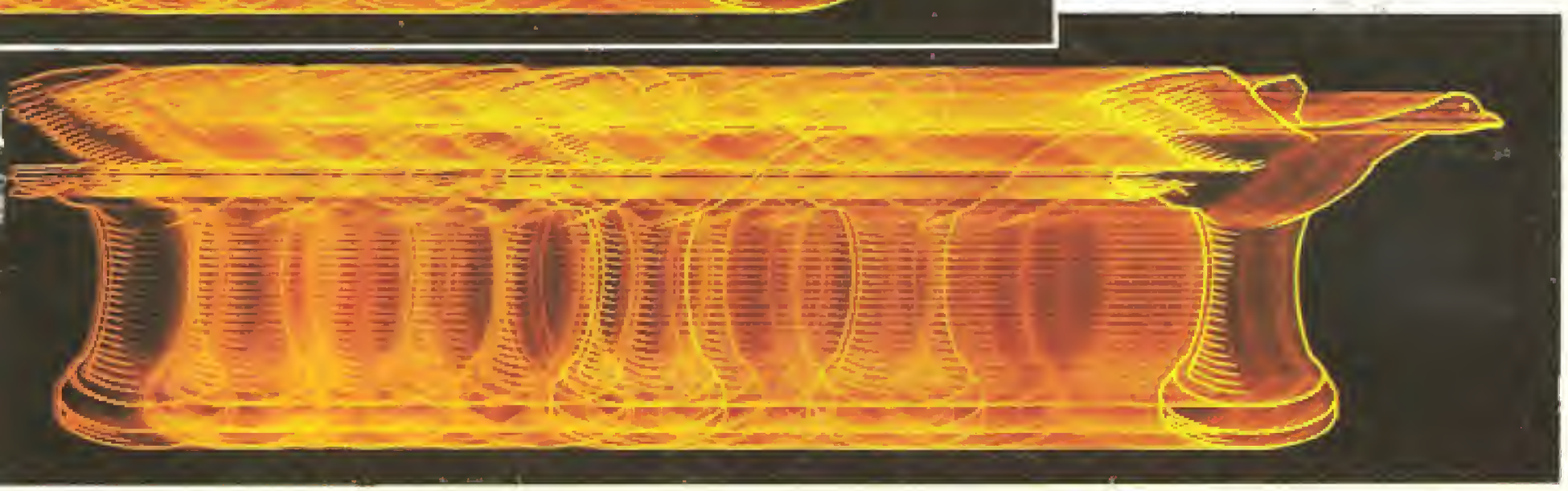
VISION GLOBAL

El programa trabaja evaluando cada una de las posiciones del juego de acuerdo con la configuración de las fichas. A cada posición se le adjudica un valor numérico en el programa, de este modo, el programa es capaz de seleccionar el mejor movimiento verificando en los resultados el valor más alto.

El programa opera en tres niveles cuando planea la jugada. En el nivel uno sólo proyecta un movimiento, se trata del así denominado *un movimiento*. En el nivel superior del juego utiliza el algoritmo alfa-beta para ahorrar tiempo al buscar entre las muchas vías de posibilidades que progresivamente se van multiplicando. En niveles intermedios, el programa repasa todas las posibilidades existentes.

Las rutinas desde la línea 2010 hasta la línea 3000 sólo se ejecutan una vez, de ahí que estén ubicadas al final del programa. Con estas rutinas *de poco uso* situadas al final del programa, las rutinas principales pueden ser ubicadas cerca del principio del programa para lograr mayor velocidad.





INICIALIZACION

He aquí las rutinas para inicializar el juego. Las direcciones han sido dimensionadas y las funciones definidas. Estas líneas definen el cuadro de gráficos:

```
2Ø1Ø  DIM G(4): DEF FN
      U(A)=INT (A-4*INT (A/4)):
      DEF FN V(A)=INT (A-8*INT
      (A/8))>=4: DEF FN
      W(A)=INT (A-2*INT (A/2))
2Ø15  LET HF=Ø: LET HG=Ø
2Ø2Ø  DIM B(32): LET B(1)=1:
      FOR I=1 TO 31: LET
      B(I+1)=B(I)*2: NEXT I
2Ø26  LET BX=B(32)*2-B(25):
```

```
      LET E=1E3Ø: LET
      H=-1E3Ø
2Ø3Ø  LET L2=LN (2)
2Ø4Ø  DIM B$(16,2,16): DIM
      G$(2,4): LET G$(1)="  ":
      LET G$(2)=" "+CHR$
      146+CHR$ 147: DIM
      H$(2, 4): LET H$(1)="  ":
      LET H$(2)=CHR$
      146+CHR$ 147+" "
2Ø5Ø  LET X=1: FOR A=1 TO 2:
      FOR B=1 TO 2: FOR C=1
      TO 2: FOR D=1 TO 2: LET
      B$(X,1)=G$(D)+G$(C)+
      G$(B)+G$(A)
2Ø6Ø  LET B$(X,2)=H$(A)+H$
      (B)+H$(C)+H$(D):LETX=
      X+1:NEXTD:NEXTC:NEXT
      B:NEXTA
2Ø7Ø  DIM S$ (8,16): GO SUB
      6ØØØ
2Ø9Ø  DIM F$(2,4): LET
      F$(1)="  "+CHR$
      144+CHR$ 145: LET
      F$(2)=CHR$ 144+CHR$
      145+"  "
2Ø95  DEF FN C(B)=FN U(B-
      1)*(4-8*FN V(B-
      1))+12*FN V(B-1)
```

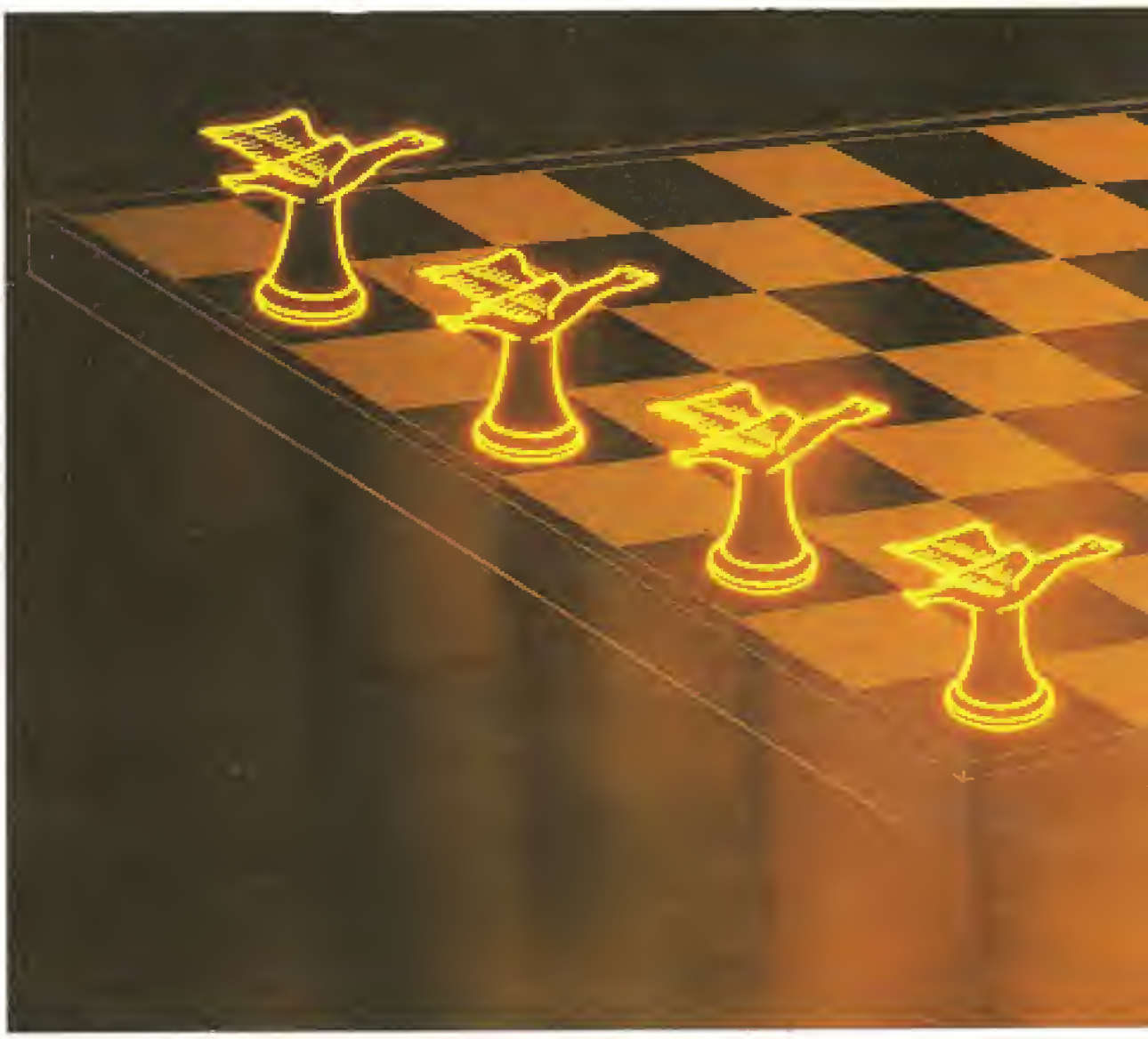
La línea 2010 dimensiona la dirección utilizada para almacenar las posiciones de las ocas. La línea 2020 numera cada casilla del tablero que se utilizará en el juego.

La línea 2030 determina el número de configuraciones que pueden ser evaluadas por el programa. 0.001 ha sido añadido cuando se ha definido la función A para evitar el redondear errores cuando se utilizan logaritmos. La dirección B$, dimensionada en la línea 2040, se utiliza para mostrar líneas del tablero que están ocupadas por piezas. F$ se utiliza para la pieza que representa al zorro y a la casilla blanca, y H$ para las ovejas y las casillas blancas. S$ se utiliza para determinar los números de las casillas.

AL INICIO

Esta rutina permite al jugador elegir quién juega, qué piezas (si el zorro o las ocas), y seleccionar el nivel de habilidad del ordenador. ¡No es demasiado difícil —al principio—!

```
27ØØ  LET F=2: LET G(1)=29:
      LET G(2)=3Ø: LET
      G(3)=31: LET G(4)=32:
      GO SUB 271Ø: GO TO
      1Ø1Ø
271Ø  CLS : PRINT AT Ø,9; INK
      1;"ZORROS Y OCAS":
      INPUT "QUIERES...";TAB
      5;"JUGAR A SER ZORRO?
      (S/N)";I$
272Ø  LET PF=Ø: IF I$="S" OR
      I$="s" THEN GO TO 276Ø
273Ø  LET PF=1: IF I$<>"N"
      AND I$<>"n" THEN GO
      TO 271Ø
274Ø  INPUT "NIVEL DE
      DIFICULTAD DEL
      ZORRO? ";SF: IF SF<1 OR
      SF>1Ø THEN GO TO 274Ø
275Ø  LET HF=131*(SF=5)+613
      *(SF=6)+1997*(SF)>6)
276Ø  INPUT "QUIERES...";TAB
      6;"JUGAR A SER OCA? (S/
      N)";I$
277Ø  LET PG=Ø IF I$="S" OR
      I$="s" THEN GO TO 286Ø
278Ø  LET PG=1: IF I$<>"N"
      AND I$<>"n" THEN GO
      TO 276Ø
279Ø  INPUT "NIVEL DE
      DIFICULTAD DE LAS
      OCAS? ";SG: IF SG<1 OR
      SG>1Ø THEN GO TO
      279Ø
28ØØ  LET HG=131*(SG=5)+
      613*(SG=6)+1997*
      (SG>6): IF HF<HG THEN
      LET HF=HG
286Ø  INPUT "QUIERES ALTERAR
      LA POSICION INICIAL?";
      I$: IF I$="N" OR I$=
      "n" THEN GO TO 3ØØØ
288Ø  IF I$<>"S" AND I$<>"s"
      THEN GO TO 286Ø
289Ø  GO SUB 21Ø: GO SUB
      31Ø: INPUT "QUIERES
      MOVER AL ZORRO? ";I$
29ØØ  IF I$="N" OR I$="n"
```

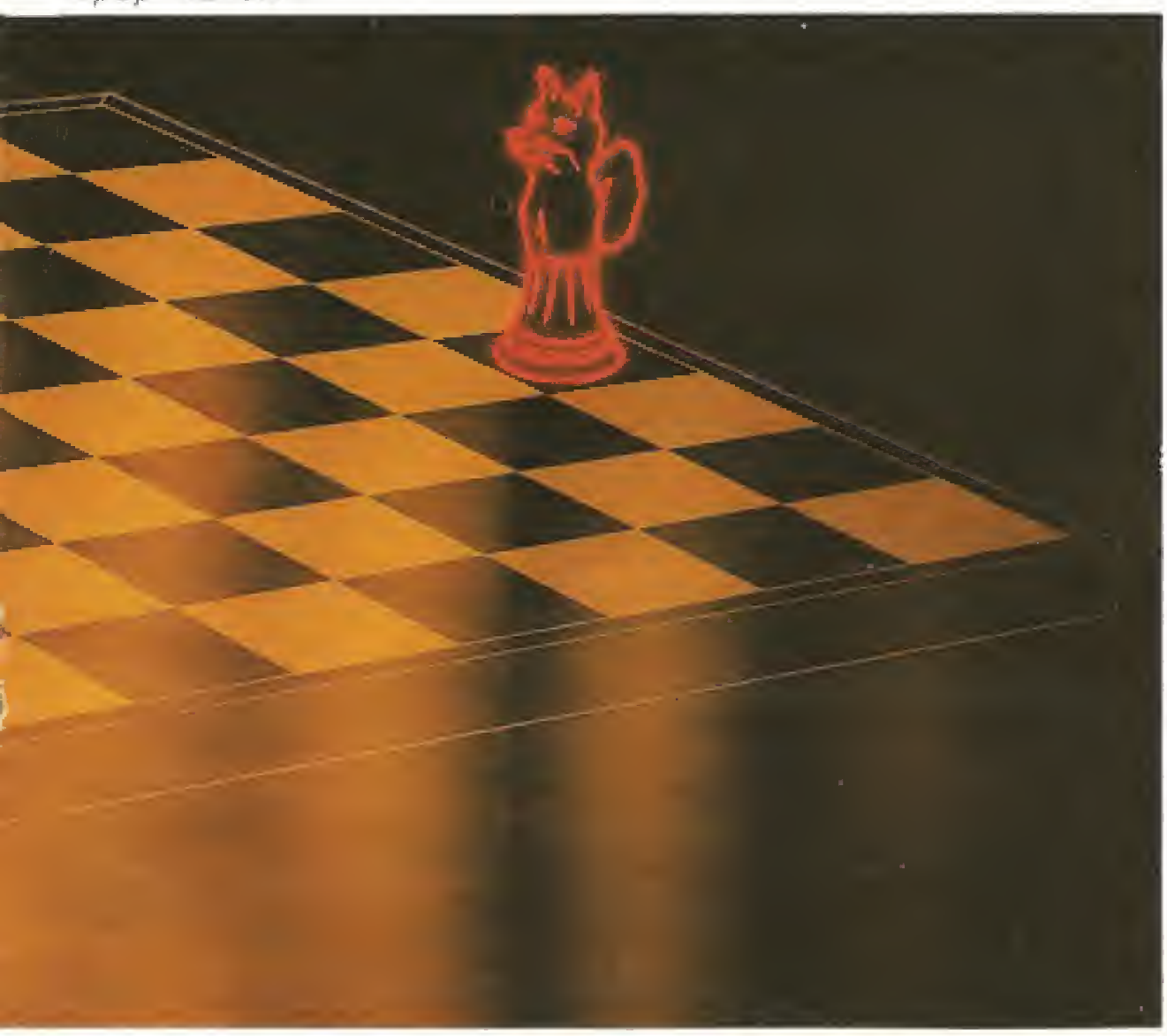


```
      THEN GO TO 2930
2910  IF I$<>"S" AND I$<>"s"
      THEN GO TO 2890
2920  INPUT "MOVER EL ZORRO
      A ";F: IF F<1 OR F>32
      THEN GO TO 2920
2930  FOR G=1 TO 4: GO SUB
      210: GO SUB 310
2940  INPUT "QUIERES MOVER
      LA OCA";(G(G));"?";I$
2950  IF I$="N" OR I$="n"
      THEN GO TO 2990
2960  IF I$<>"S" AND I$<>"s"
      THEN GO TO 2940
2970  INPUT "MOVER LA OCA A
      ";I: IF FN X(I) OR I=F
      THEN GO TO 2960
2972  IF I<1 OR I>32 THEN GO
      TO 2970
2980  LET G(G)=I
2990  NEXT G: IF FN X(F) THEN
      PRINT "HAY UNA OCA
      BAJO EL ZORRO": FOR
      I=1 TO 1500: NEXT I
3000  RETURN
4000  PRINT AT 21,0;"
4010  RETURN
```

La línea 2700 determina la posición de partida, con las cuatro ocas ocupando las cuatro casillas de la parte inferior del tablero, y el zorro ocupando la segunda casilla de la línea superior izquierda del tablero.

Una vez que la línea 2700 ha inicializado la posición de partida del zorro y las ocas, las líneas 2710 y 2750 dan al jugador la opción de jugar a ser el zorro y determina quién va a jugar a ser el zorro. Las líneas 2760 a 2800 son similares, sólo que aquí al jugador se le da la opción de jugar a personificar las ocas.

El juego ha sido diseñado de manera que permite, o bien ajustar la posición de partida, o bien continuar en la posición de partida en la que dejaste el juego la última vez (necesitarás tomar notas de las posiciones de las fichas cuando acabe el juego) o bien intentar ganar (¡o perder!) desde una posición particularmente interesante. Las líneas 2860 a 3000 preguntan si el jugador desea modificar la posición de partida, da la señal de *preparado* y asegura que las posiciones elegidas sean correctas.



TRAZANDO MOVIMIENTOS

La rutina de trazado de movimientos es una de las más importantes del programa.

```
140  DEF FN X(B)=B=G(1) OR
     B=G(2) OR B=G(3) OR
     B=G(4)
2100 DIM R$(8,16)
2142 DEF FN Z(B)=(B=G(1))+(
     B=G(2))*2+(B=G(3))*3+
     (B=G(4))*4
2150 DIM M(4,32): DIM X(32):
     DIM Z(32)
2160 FOR B=1 TO 32: LET
     U=B-1-4*INT (B/4-.2):
     FOR A=1 TO 4: LET M(A,
     B)=(B,2)-2*U+8*((B<5)
     OR(A>2))+(A*7-6)*(U=
     3)+(A=2)+(A=4): NEXT
     A: LET X(B)=((B>4)+
     (B<29))*((U<3)+1):
     LET  Z(B)=(B>4)*((U<3)
     +1): NEXT B
2180 DIM V(11): DIM A(11):
     DIM F(11): DIM P(11):
     DIM C(11): DIM R(1): DIM
     S(1)
```

¿OTRA PARTIDA?

Añade a continuación otra rutina: «¿Otra partida?»

```
141  INPUT "OTRO JUEGO (S,
     N) ?";I$
1420 IF I$="S" OR I$="s"
     THEN GO TO 2700
1430 IF I$<>"N" AND I$<>"n"
     THEN GO TO 1410
1440 STOP
```

Estas líneas deberían resultarnos ya familiares, y entran en acción cuando las ocas intentan atrapar al zorro, o cuando el zorro intenta alcanzar el lado opuesto del tablero.

No intentes ejecutar el programa con RUN en esta fase, pues aún hay que añadirle muchas partes de vital importancia. En la siguiente parte del artículo insertarás las rutinas que te permitirán ejecutar el juego.

EL ZORRO Y LAS OCAS (Y III)

Deja que tu ordenador recoja ideas antes de comenzar el juego. Con estas rutinas podrá jugar el papel del zorro o de las ocas, y adelantarse para improvisar su jugada.

Ésta es la tercera y última parte del artículo «El juego del zorro y las ocas». Las rutinas que quedan son las que permitirán al ordenador jugar la parte, ya sea del zorro o de las ocas. De hecho se puede programar al ordenador para que juegue ambas partes a la vez, cuando compita contra sí mismo, o bien puedes programarlo para que no juegue a ser ninguna de las figuras, y jugar tú contra algún amigo utilizando el ordenador en lugar de un tablero y fichas.

Éstas son dos de las rutinas más importantes del programa. La rutina que comienza en la línea 210 hasta la línea 230 evalúa la posición de juego resumiéndola en un número de una sola cifra. Luego, una vez que el programa ha decidido el valor del mejor movimiento, ha de convertir ese número en un movimiento o en un juego de posiciones para las fichas. Las líneas 250 a 260 transforman el valor simple en el juego de valores necesarios para posicionar las fichas.

Estas subrutinas son llamadas con frecuencia durante el programa, de manera que deberán estar colocadas correctamente al inicio del programa.

```
21Ø LET P=B(G(1))+B(G(2))+B
    (G(3))+B(G(4))
22Ø LET X=F: IF P<B(32) THEN
    LET P=P-BX: LET X=
    33-F
23Ø LET P=P*B(X):
    RETURN
25Ø LET F=FN A(ABS P)-3Ø:
    LET B=P/B(F): IF B<Ø
    THEN LET B=B+BX: LET
    F=33-F
26Ø FOR A=1 TO 4: LET
    G(A)=FN A(B)+1: LET
    B=B-B(G(A)): NEXT A:
    RETURN
```

EL MOVIMIENTO DEL ZORRO

Las líneas 1020 a 1180 dirigen el movimiento del zorro. La rutina utiliza la rutina de visualización del tablero para mostrar el estado actual del tablero, seguidamente pasa a comprobar si el zorro ha ganado en la línea 1030. En la línea 1032 comprueba si hay allí, al menos, un movimiento legal que pueda realizar el zorro (de lo contrario, ganan las ocas).

Si el jugador es quien lleva el control de las ocas, las líneas 1050 a 1070 se encargan de introducirlo en el ordenador y de comprobar que el movimiento efectuado sea efectivamente correcto. Si, por otra parte, es el ordenador el que controla al zorro, las líneas 1110 a 1112 son las encargadas de llevar a cabo el movimiento. El número de pliegues que el programa observa, M, y el pliegue actual considerado, L, son colocados en la línea 1110. Las líneas 1120 a 1180 son una subrutina que evalúa el mejor movimiento.

Se utilizan cuatro direcciones en la rutina para el mejor movimiento: A contiene el número de movimientos que aún pueden ser intentados; V, el mejor resultado hasta el momento; P, el movimiento que ha conseguido este resultado; y F, la posición previa del zorro.

```
1Ø1Ø GO SUB 21Ø
1Ø2Ø GO SUB 31Ø: GO SUB 25Ø
1Ø3Ø IF F>28 THEN PRINT AT
     21,Ø;"EL ZORRO HA
     GANADO": GO TO 141Ø
1Ø32 GO SUB 41Ø: IF V=H
     THEN PRINT "LAS OCAS
     HAN GANADO": GO TO
     141Ø
1Ø4Ø IF PF THEN GO TO 111Ø
```

LOS MEJORES DE INPUT

MAYO 1987

sinclair

PUESTO	TITULO	PORCENTAJE
1.º	Green Beret	20,4 %
2.º	Commando	19,5 %
3.º	Army Moves	11,2 %
4.º	Saboteur	10,2 %
5.º	Antiriad	8,3 %
6.º	Ghost'n Goblins	7,5 %
7.º	Trivial Pursuit	6,4 %
8.º	The Great escape	6,4 %
9.º	El Misterio del Nilo	5,5 %
10.º	Gauntlet	4,6 %
		100,0 %

ELIGE TUS PROGRAMAS

Hemos pensado que es interesante disponer de un *ranking* que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en INPUT.

El resultado de las votaciones será publicado en cada número de INPUT.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «LOS MEJORES INPUT».

Enviad vuestros votos a: LOS MEJORES DE INPUT Aribau, 185. Planta 1. 08021 Barcelona

SIN. N.º 19

1er Título elegido

2º Título elegido

3er Título elegido

Programa que te gustaría conseguir

Qué ordenador tienes

Nombre

1er Apellido

2º Apellido

Fecha de nacimiento

Teléfono

Dirección

Localidad

Prov.

LAS RUTINAS DE GRABACION

Seguramente el grupo de las rutinas de control del cassette están entre las de mayor uso e importancia del sistema operativo residente en la ROM. Como ya sabrás, llevan a cabo todas las tareas de carga, grabación, verificación y mezcla de programas, pero no mucha gente sabe cómo sacar más partido a esta serie de rutinas cuando se trabaja desde código máquina.

En este artículo intentaremos desvelar gran parte de esos secretos a los que siempre has intentado dar una respuesta (el por qué de las cabeceras falsas, rutinas de carga y grabación con colores inusuales, bloques grabados con distinto tono, lectores de cabeceras, etc...).

Todos esos temas tienen su origen en una particular manipulación de las rutinas ubicadas en ROM, y muchas veces se requiere un simple conocimiento de la constitución de dichas rutinas para modificar los bytes necesarios y así conseguir el efecto deseado. No siempre estamos obligados a diseñar una nueva rutina de carga o grabación, sino simplemente saber utilizar correctamente la maravillosa «caja de herramientas» que es nuestro sistema operativo (así, muchas veces sólo necesitaremos generar una copia de la rutina de la ROM en otra parte de la memoria, modificar unos pocos datos, y ejecutarla en su nueva posición).

FUNCIONAMIENTO GENERAL DE LAS RUTINAS

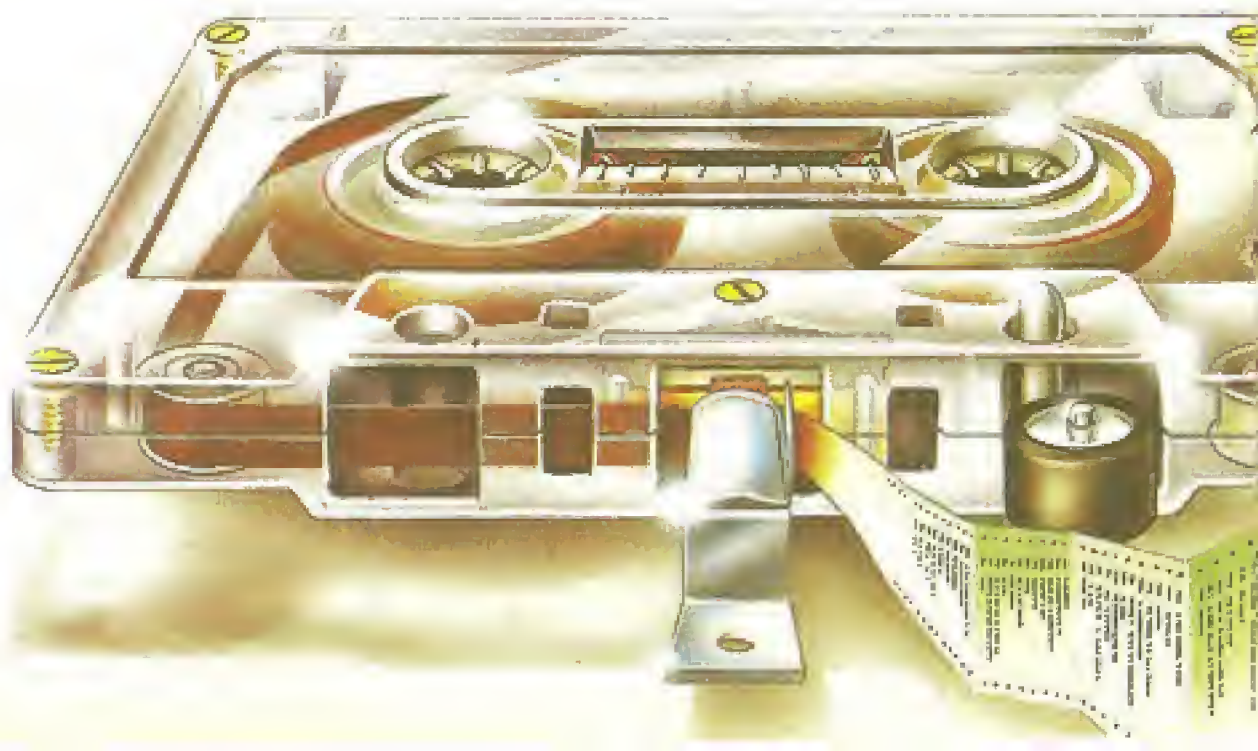
Las rutinas de entrada/salida del cassette constituyen uno de los medios más normales y baratos que tenemos para comunicarnos con el medio exterior. Pero ¿cómo se lleva a cabo esta comunicación a nivel de código máquina? No queremos aburrirte con una serie de complejos mecanismos de control mediante software a nivel de sistema operativo, pero sí reseñar que fundamentalmente nuestro ordenador envía y recibe información del cassette por medio de los port de salida, y más concretamente utilizando el port 254 (#FE en hexadecimal), que, como comprobarás, es el mismo que utiliza la ULA para gestionar los cambios del color del borde, o incluso el control del teclado. Los bits D0, D1 y D2 son los dedicados para el color del borde, pero los que nos interesan en este momento son los D6 (conexión EAR de nuestro cassette) y D3 (conexión MIC). El bit D4 es utilizado para el control del altavoz.

Mediante una serie de bucles más o menos complejos, se van almacenando en los registros adecuados (suele ser el HL), el número de estados (o impulsos digitales) que se van recibiendo o se pretende enviar al cassette. Así, poco a poco, se recopila una colección de bits que más tarde compondrán nuestros bytes a transmitir entre la ULA y nuestro cassette. Mediante los impulsos adecuados vamos controlando las salidas EAR y MIC de nuestro cassette, y así conseguiremos la carga o grabación, según sea el caso.

La velocidad de transmisión de datos se mide en Kbits por segundo o baudios, siendo la de nuestro Spectrum de 1500 baudios, una velocidad lo suficientemente rápida como para no «eternizar» la carga y, además, ofrecer una cierta seguridad en la transmisión de datos (en otros equipos, como los MSX, existen dos velocidades distintas de grabación, una inferior y otra superior a la de nuestro Spectrum). Las famosas rutinas «turbo» sacrifican una parte de seguridad en la transmisión de datos, en aras de conseguir una mayor velocidad.

LAS RUTINAS DE GRABACION

Su manejo nos permite enviar bloques de datos al cassette de una manera rápida y sencilla, y nos referimos a bloques de bytes solamente porque normalmente es nuestro objetivo cuando trabajamos en código máquina. Sería un poco ilógico tratar de grabar un programa BASIC desde C/M, no porque no se pueda hacer (básicamente sería grabar dos bloques, uno con la información de la cabecera, y otro con el programa en sí, como hace el sistema operativo), sino



- FUNCIONAMIENTO GENERAL DE LAS RUTINAS
- LAS RUTINAS DE GRABACION
- MANIPULANDO LAS RUTINAS DE GRABACION

porque después de hacer una rutina en C/M que nos preguntara el nombre y lo almacenara en las direcciones de memoria adecuadas junto con ciertos valores necesarios de las variables del sistema, nos daríamos cuenta que es mucho más sencillo y casi igual de rápido usar la instrucción ¡¡¡SAVE!!!.

La rutina principal y de más fácil uso es la ubicada en la dirección de memoria #04C2. Con ella conseguiremos grabar un bloque de bytes de la memoria a partir de una dirección especificada. Antes de ejecutarla introduciremos la información del bloque a grabar en los diferentes registros:

— Cargaremos el acumulador con #0 para indicar que queremos grabar los datos de una cabecera, o con #FF y así grabaremos un bloque de bytes.

— En el registro IX introduciremos la dirección de comienzo en la memoria de la cabecera o bloque de bytes a grabar.

— En el registro doble DE almacenamos la longitud en bytes de los datos a grabar.

A modo de ejemplo, supongamos que estamos interesados en grabar en cassette 1024 bytes del principio de la memoria, para conseguirlo deberemos ejecutar la minirrutina:

```
10 ORG  60000   ;Ubicamos nuestra rutina en dicha posicion
11 ENT  60000   ;Posicion de ejecucion
12 LD   A,#FF   ;Para grabar un bloque de bytes
13 LD   IX,#00  ;de la direccion de memoria 0
14 LD   DE,1024 ;de longitud 1 Kb
15 CALL #04C2   ;Rutina de grabacion
16 RET          ;El necesario retorno al BASIC.
```

Aquellos que no dispongan de un programa ensamblador pueden utilizar el siguiente programa cargador en BASIC, que cumplirá el cometido a la perfección.

```
10 CLEAR 59999: FOR N=0 TO
   12: READ A: POKE
   (60000+N),A:
   NEXT N
20 DATA 62,255,221,33,0,
   0,17,0,4,205,194,4,
   201
```

Para ejecutar la rutina pon en marcha el cassette para grabar y teclea RANDOMIZE USR 60000. Se ha de hacer notar que el bloque de bytes así grabado no se puede recuperar con un LOAD normal desde el BASIC, sino que ha de hacerse de una manera, aunque sencilla, diferente. Para ello deberemos utilizar desde código máquina, la rutina de carga de bloques de bytes #0556.

Esta rutina de carga de bloques de bytes #0556 la describiremos detalladamente cuando abordemos el tema de las rutinas de carga.

No obstante, como adelanto y en aras de la claridad de exposición, hemos creído preferible incluirla a continuación:

```
1Ø ORG 5ØØØØ    ;Lugar de ubicacion de la rutina
2Ø ENT 5ØØØØ    ;Posicion de ejecución
3Ø SCF          ;Set Carry Flag, indica que vamos a cargar
4Ø LD A, #FF    ;Para cargar un bloque, tono 255
5Ø LD IX, 4ØØØØ ;Lo cargamos a partir de la 4ØØØØ
6Ø LD DE, 1Ø24  ;Carga un bloque de longitud 1 Kb
7Ø CALL #Ø556   ;Llamada a la rutina
8Ø RET          ;Retorno al BASIC
```

El cargador BASIC quedará:

```
1Ø CLEAR 49999: FOR N=Ø TO
   13: READ A: POKE
   (5ØØØØ+N),A: NEXT N
2Ø DATA 55,62,255,221,33,64,
   156,17,Ø,4,2Ø5,86,5,2Ø1
```

Todo este método de proceder implica una serie de desventajas, pero también muchas facilidades. Cuando seguimos este método de grabación, sólo nosotros conocemos el contenido y longitud exacta de lo que hay grabado, por lo que aunque puedan copiar nuestros programas, si no tienen la rutina de carga precisa, nos evitaremos muchas miradas curiosas...

Como el lector podrá comprobar, la rutina no es nada espectacular, pero puede servirnos como primer punto de contacto con el tema. También podrá observar que al ejecutar la rutina no aparece el habitual mensaje «Star tape, then press any key», lo cual indica que hemos interceptado el sistema operativo en un punto posterior al normalmente utilizado por el ordenador. Si desea simular también este efecto, deberá añadir antes de las anteriores, las líneas que utiliza el sistema operativo para tal efecto (eliminando las líneas 100 ORG 60000 / 110 ENT 60000):

```
1Ø ORG  6ØØØØ      ;Direccion de almacenaje de la rutina
2Ø ENT  6ØØØØ      ;Direccion de ejecucion
3Ø LD   A,#FD      ;Esta linea y la siguiente son utilizadas
4Ø CALL #1601      ;para abrir el canal 'K'
5Ø XOR  A          ;Carga A con la posicion del mensaje (la
6Ø LD   DE,#Ø9A1   ;primera) dentro de una tabla.
7Ø CALL #ØCØA      ;Imprime el mensaje seleccionado
8Ø SET  5,(IY+2)   ;Activa TV-FLAG para borrar
9Ø CALL #15D4      ;Espera a que se pulse una tecla
```

Por tanto, nuestro cargador BASIC quedará:

```
1Ø CLEAR 59999: FOR N=Ø TO
   31: READ A: POKE
   (6ØØØØ+N),A: NEXT N
2Ø DATA 62,253,2Ø5,1,22,175,
   17,161,9,2Ø5,1Ø,12,253,
   2Ø3,2,238
3Ø DATA 2Ø5,212,21,62,255,
   221,33,Ø,Ø,17,Ø,4,2Ø5,
   194,4,2Ø1
```

Ahora, al ejecutar RANDOMIZE USR 60000 veremos cómo aparece el mensaje y espera a que pulsemos una tecla antes de grabar en cassette. Como tampoco hemos grabado la cabecera, necesitamos seguir utilizando la rutina de carga de bytes ya descrita.

Hasta ahora hemos utilizado la rutina principal, pero en ROM también se encuentran, por así decirlo, «los pasos anteriores», que son las rutinas de control de la que hemos utilizado. Cuando ejecutamos SAVE, se graban la cabecera y el bloque de bytes correspondiente, no sólo este último.

Antes de seguir deberemos definir el concepto de cabecera. Básicamente podemos decir que se trata de un bloque de 17 bytes grabados con tono 0 (LD, A, 0), conteniendo la información de lo que se pretende grabar:

— El primer byte de la cabecera indicará el tipo de programa a grabar, siendo 0 para un programa Basic, 1 para una matriz numérica, 2 alfanumérica, y 3 para código máquina.

— Los siguientes 10 bytes contendrán el nombre.

— En seguida se alojan 2 bytes indicando la longitud del bloque a grabar.

— Los dos siguientes bytes nos indicarán la dirección de autoejecución si se trata de un programa Basic, o la situación en la memoria en caso de ser un bloque en código máquina.

— Los datos almacenados en los bytes 16 y 17 son utilizados en caso de grabar un programa Basic y reflejan el contenido de la variable VARS.

Si hemos grabado una cabecera con datos correctos y a continuación un bloque concordante con lo allí almacenado, comprobaremos cómo podremos realizar una carga posterior de lo grabado en cinta mediante el comando Basic LOAD o incluso desde código máquina. P. ej., si quisiéramos grabar el contenido de una pantalla en cassette, para después volver a recuperar su contenido mediante LOAD "" CODE, deberemos proceder de la manera siguiente:

— Crearemos una cabecera con los siguientes datos:

1. Primer byte 3 para indicar código máquina.

2. A continuación almacenaremos el nombre, el cual ocupará como máximo 10 caracteres, y si no se llega a este número se deberán dejar espacios en blanco para rellenar el espacio.

3. Los dos siguientes bytes (el 11 y 12 suponiendo que el primer byte es el cero) almacenan la longitud, en este caso 6912 (longitud del archivo de pantalla más el de atributos), por tanto el byte bajo será 0 y el alto (el 12) será 27.

4. La dirección de donde extraemos los datos la contendrán los bytes 13 y 14, y en este caso será 16384 (dirección de comienzo del archivo de pantalla), por tanto el byte 13 deberá contener un 0, y el byte 14 un 64.

Por tanto, nuestra rutina de grabación quedará como:

```
1Ø      ORG  6ØØØØ
2Ø      ENT  6ØØØØ
3Ø      LD   HL, (VALOR+13)
4Ø      LD   IX,VALOR
5Ø      CALL #Ø97Ø
6Ø      RET
```

```
7Ø VALOR  DEFB 3
8Ø        DEFM"PANTALLA  "
9Ø        DEFB Ø,27,Ø,64
```

Los bytes 16 y 17 no reciben ningún valor porque no se utilizan en el caso de grabación de bloques de código máquina. Tenga en cuenta que después del nombre dado de PANTALLA (8 letras), deberá dejar dos espacios en blanco para no falsear los valores siguientes.

En la línea 20 guardamos la dirección de donde vamos a sacar los datos, ya que esta rutina (la #0970) así lo requiere. En un principio cargamos IX con la dirección de almacenamiento de los datos de la cabecera en la memoria, ya que así como la rutina #04C2 distinguía entre grabar una cabecera o un bloque (indicándoselo en A), la rutina #0970 gestiona automáticamente un proceso en el que primero se graban 17 bytes de la dirección apuntada por IX (los datos de la cabecera), y luego según la información aportada por estos datos, va cargando los diferentes registros con los valores de esta cabecera para llamar a continuación a la rutina #04C2 y grabar el bloque. Es decir, lleva a cabo a labor que tuvimos que hacer anteriormente, y después de grabar la cabecera (LD A,0 / LD IX, VALOR / LD DE, #0011 / CALL #04C2), lleva a cabo la grabación del bloque según sus datos (LD A, #FF / LD E,(IX+11) / LD D,(IX+12) / LD IX,(VALOR+13) / CALL #04C2). El cargador BASIC de la rutina podría ser:

```
1Ø CLEAR 59999: FOR N=Ø TO
   25: READ A: POKE
   (6ØØØØ+N),A: NEXT N
2Ø DATA 42,12Ø,234,221,33,
   1Ø7,234,2Ø5,112,9,2Ø1,3
3Ø DATA 8Ø,65,78,84,65,76,
   76,65,32,32
4Ø DATAØ,27
5Ø DATAØ,64
```

En el cargador BASIC la línea 30 almacena los caracteres ASCII de la palabra "PANTALLA ", la línea 40 la longitud del bloque a grabar (6912 dividido en sus dos bytes) y la 50 almacena la dirección donde empieza el bloque, en este caso la 16384. Con estos datos el lector podrá crear sus propias cabeceras y hacer sus pruebas de grabación de bloques de datos con cabeceras. Por ejemplo, después de rodar el cargador Basic, ejecute la línea de prueba:

```
FOR N=1 TO 704: PRINT PAPER
(INT (7*RND)); INK 9;"#";: NEXT
N: RANDOMIZE USR 60000
```

Si comprueba el resultado del ejemplo expuesto, observará que ha grabado la pantalla, la cual será fácilmente recuperable con LOAD "PANTALLA"SCREEN$.

En la ROM (Read Only Memory o Memoria de sólo lectura), como su nombre indica, no podemos grabar datos o alterar alguno de los ya existentes, lo cual es una buena medida de seguridad, pero también limita nuestra libertad para intentar mejorar o adaptar a nuestras necesidades alguna de las rutinas allí existentes. También es cierto que la ROM de nuestro Spectrum está bastante lograda, y es muy difícil que tengamos necesidad de cambiarla sistemáticamente, lo cual no quita para que nosotros podamos modificar una copia de estas rutinas realizada en la RAM.

En el caso que nos ocupa, una vez conocida la longitud y disposición en memoria de las rutinas de grabación de la ROM, podemos realizar una copia de estas rutinas en la RAM y modificarlas a nuestra conveniencia. El fin más sencillo y vistoso que podemos conseguir es modificar el proceso de grabación para que se realice con otros colores en el borde (como en muchos programas comerciales).

Principalmente, las dos rutinas que necesitamos copiar son las que hemos estado manejando, la #0970 y la #04C2. La primera de ellas realizaba el control del proceso grabando primero la cabecera, y luego según ésta, el bloque de bytes (seguramente has reparado en que el orden en la memoria parece el contrario al que sería el lógico). La manera más rápida de copiarlas es mediante el comando de C/M LDIR:

— La rutina #0970 (2416 en decimal) ocupa 49 bytes, por tanto, llega hasta la posición #09A0 (2462 en decimal).

— La rutina #04C2 (1218 en decimal) ocupa 125 bytes, y se extiende hasta la posición de memoria #053E (1342 en decimal).

— Si queremos organizar nuestra rutina a partir de la posición 60000, deberemos copiar los dos bloques en las posiciones 60007 y 60071 respectivamente, para dejar espacio para copiar antes de la primera rutina las líneas:

```
1Ø ORG  6ØØØØ
2Ø ENT  6ØØØØ
3Ø LD   HL,(VALOR+13)
4Ø LD   IX,VALOR
```

El espacio dejado antes de la segunda rutina servirá para alojar los datos de la cabecera que queremos grabar:

```
26Ø VALOR  DEFB3
27Ø        DEFM "PANTALLA"
28Ø        DEFB Ø,27,Ø,64
```

La cabecera utilizada corresponde a los datos usados en la rutina anterior (grabación de una pantalla) para que el lector pueda seguir mejor la idea expuesta.

— Por tanto, el proceso a seguir será:

1. Copiar los dos bloques. Para el primero utilizaremos la rutina siguiente, en la que la línea 30 nos indica de dónde vamos a copiar, la línea 40 nos señala dónde lo vamos a llevar, y la línea 50, el número de bytes a copiar:

```
1Ø ORG  4ØØØØ
2Ø ENT  4ØØØØ
3Ø LD   HL,#Ø97Ø
4Ø LD   DE,6ØØØ7
5Ø LD   BC,49
6Ø LDIR
7Ø RET
```

o bien ejecutaremos la rutina con el cargador Basic:

```
1Ø CLEAR 39999: FOR N=Ø TO
   11: READ A: POKE
   (4ØØØØ+N),A: NEXT N
2Ø DATA 33, 112, 9,17,1Ø3,
   234,1,49,Ø,237,176,2Ø1
```

Una vez rodado, lo ejecutaremos con RANDOMIZE USR 40000. Para

el segundo bloque cambiaremos las líneas 30, 40 y 50 del listado ensamblador por:

```
3Ø LD HL,#Ø4C2
4Ø LD DE,6ØØ71
5Ø LD BC,125
```

o también cambiar la línea de DATA's del cargador:

```
2Ø DATA 33, 194,4,17,167,
   234,1,125,Ø,237,176,2Ø1
```

y rodarlo con la nueva modificación ejecutando de nuevo RANDOMIZE USR 40000.

2. Si estamos trabajando con un programa ensamblador-desensamblador, deberemos desensamblar desde la posición de memoria 60007 hasta la 60195, pasar el texto desensamblado al ensamblador, y añadir las 4 líneas y la información de la cabecera (líneas 10 a 40 y 260 a 280) a las que antes hacíamos referencia, en los huecos de memoria reservados.

NOTA: Para desarrollar este artículo se ha trabajado con el paquete ensamblador-desensamblador DYPACK de Hisoft, que ofrece la facilidad antes mencionada. Igualmente hemos utilizado el símbolo '#' para referirnos a números hexadecimales, ya que este programa así lo utiliza.

Si usted no dispone de un programa de las características anteriores, deberá realizar los POKE's:

```
1Ø FOR N=6ØØØØ TO 6ØØØ6:
   READ A: POKE N,A: NEXT N
2Ø DATA 42,165,234,33,152,
   234
3Ø FOR N=6ØØ56 TO 6ØØ7Ø:
   READ A: POKE N,A: NEXT N
4Ø DATA 3,8Ø,65,78,84,65,76,
   76,65,32,32,Ø,27,Ø,64
```

— Después de este punto tendremos copiadas las rutinas de la ROM a partir de la posición 60000, pero todos los saltos y llamadas seguirán realizándose al sistema operativo, y no a nuestra rutina (por ejemplo, si ejecutamos la rutina de la posición 60000, funcionará, pero estará llamando a la #04C2 de la ROM).

— Para evitar esto deberemos «reformar» nuestras etiquetas, es decir, donde aparezca CALL #04C2 o JP #04C2 deberemos poner CALL GRABAR o JP GRABAR (suponiendo que llamemos GRABAR a la etiqueta que marca el comienzo de nuestra copia hecha de la #04C2, posición 60071). Si hemos numerado las líneas de 10 en 10 empezando por la 10, tendremos hasta la 990, y la 290 deberemos cambiarla por 290 GRABAR LD HL,#053F. Tenemos que asegurarnos que las etiquetas de salto en instrucciones como JP, JR o DJNZ deben corresponder con las etiquetas de las líneas a las que hacen referencia (hemos elegido la palabra inglesa LOOP [bucle] y un número, por resultar corta y de fácil visualización).

— Si no dispone de un programa ensamblador, introducirá estas modificaciones «a pelo». La lista de POKE's a realizar es la siguiente:

```
POKE 6ØØ34,167: POKE 6ØØ35,234: POKE 6ØØ54,167: POKE
6ØØ55,234
POKE 6Ø1Ø7,189: POKE 6Ø1Ø8,234: POKE 6Ø129,236: POKE
6Ø13Ø,234
POKE 6Ø145,1Ø : POKE 6Ø146,235: POKE 6Ø173,249: POKE
6Ø174,234
POKE 6Ø189,227: POKE 6Ø19Ø,234
```

Como antes se ha explicado, estos POKE's son debidos sobre todo a una necesidad de cambiar las direcciones de retorno en instrucciones como CALL o JP, y para que no se produzca el salto a las rutinas de la ROM, sino a nuestra propia rutina situada a partir de la 60000.

— Si usted ha seguido los pasos que hemos descrito, y la ejecuta (RANDOMIZE USR 60000), comprobará que la rutina sigue actuando como cuando lo hacía en su ubicación original, pero con una diferencia esencial: ¡Ahora es totalmente independiente de la ROM y puede ser modificada a nuestro antojo! El fin que fijamos en un principio era el de cambiar los colores del borde en la rutina de grabación, y esto se puede conseguir con una simple observación de nuestro listado.

— En la línea 390 del listado ensamblador (390 LD A,#02), la rutina carga en el acumulador el valor 2, que corresponde a enviar una señal «MIC on» y a seleccionar el color rojo (y su complementario 7-2-5, color azul claro) para las primeras líneas gruesas que aparecen en el borde. Introduzca aquí directamente el número de otro color, y el borde variará su aspecto habitual. También puede introducir directamente ese valor mediante POKE 60091,3 (hemos elegido el 3, magenta, y, por tanto, su complementario, verde).

— Para modificar el color de las líneas finas y cambiantes que vienen a continuación, puede alterar el byte bajo de la línea 570 LD BC,#3B03 sumando 8 (señal) 'MIC off' necesaria) al código del color elegido. Así, si desea volver blancas las líneas amarillas, deberá dar el valor 8+7=15 o #0F, y nuestra línea quedará: 570 LD BC, #3B0F. También puede realizar dicha operación desde el Basic con POKE 60124,15.

Podrá comprobar que a pesar del esfuerzo en diseñarla, la rutina sólo ocupa 196 bytes, y puede ser introducida en sus programas para dar un toque de curiosidad. También hay que decir que aunque sólo hemos presentado como ejemplo el tratamiento de pantallas como bloque de bytes, podemos experimentar grabando cualquier tipo de datos, con sólo variar los datos de la cabecera.

Para rodar la rutina puede introducir la línea de prueba:

```
1Ø FOR N=1 TO 44: PRINT
   PAPER (INT (7*RND)); INK 9;
   " INPUT SINCLAIR ";: NEXT
   N: RANDOMIZE USR 6ØØØØ
```

Suele ser bastante difícil modificar datos sin ver el resultado final, por eso

a continuación introducimos la rutina definitiva con todos los cambios realizados, y el cargador Basic equivalente.

Hemos conseguido nuestro objetivo, pero hay que puntualizar que aunque la rutina de grabación funciona perfectamente, si volvemos a cargar los datos grabados, estaremos utilizando de nuevo las rutinas de la ROM, y los colores de carga serán los normales. Pero eso es un tema de las rutinas de carga que estudiaremos en próximos capítulos.

Esperamos que este artículo haya servido para acercarte más al tema de las rutinas de grabación.

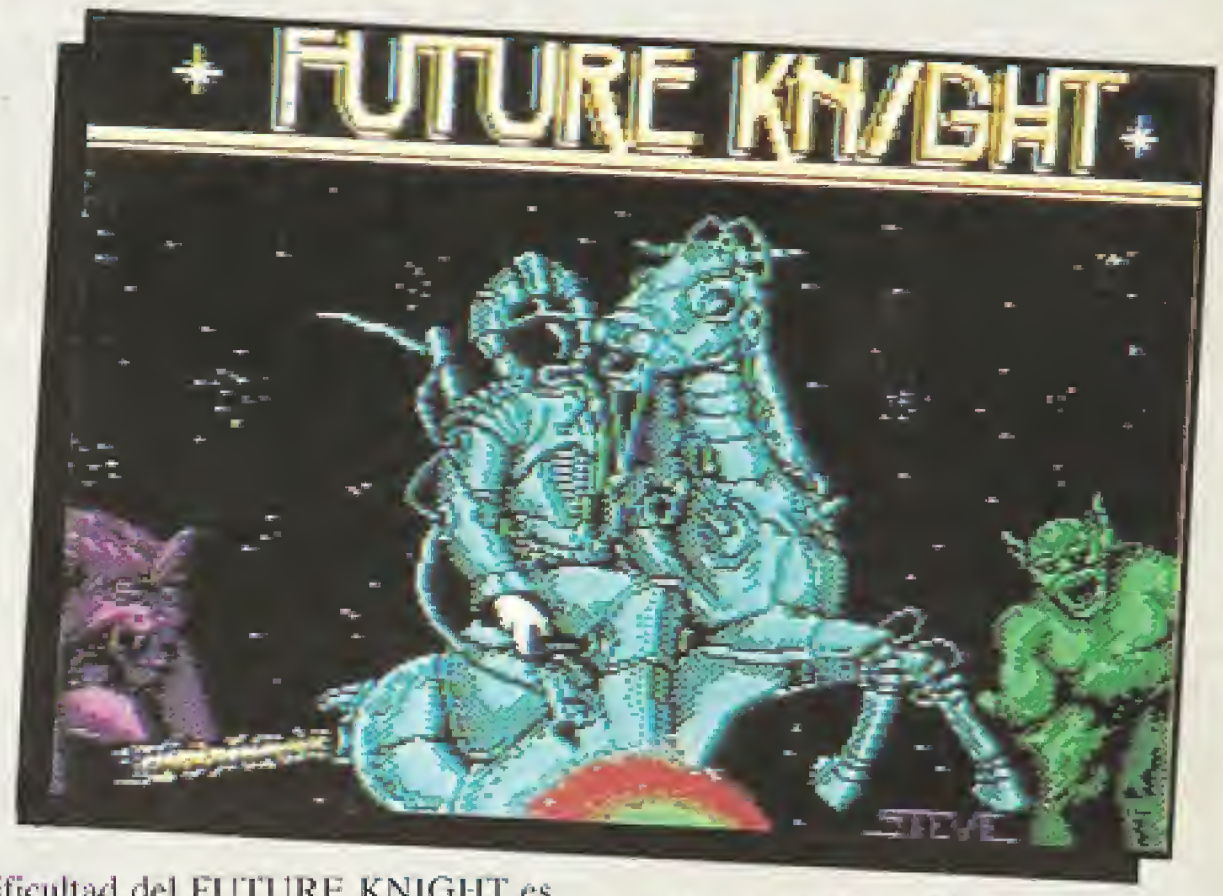
```
 1 REM **********************
 2 REM *      MOLISOFT      *
 3 REM **********************
10 CLEAR 59999
20 FOR N=60000 TO 60195:
   READ A: POKE N,A: NEXT N
30 CLS : FOR N=1 TO 44:
   PRINT PAPER (INT
   (7*RND)); INK 9;" INPUT
   SINCLAIR ";: NEXT N
40 RANDOMIZE USR 60000
100 DATA 42,165,234,221,33,
    152,234,229,62,253
110 DATA 205,1,22,175,17,
    161,9,205,10,12
120 DATA 253,203,2,238,205,
    212,21,221,229,17
130 DATA 17,0,175,205,167,
    234,221,225,6,50
140 DATA 118,16,253,221,94,
    11,221,86,12,62
150 DATA 255,221,225,195,
    167,234,3,80,65,78
160 DATA 84,65,76,76,65,32,
    32,0,27,0
170 DATA 64,33,63,5,229,33,
    128,31,203,127
180 DATA 40,3,33,152,12,8,
    19,
    221,43,243
190 DATA 62,3,71,16,254,211,
    254,238,15,6
200 DATA 164,45,32,245,5,37,
    242,189,234,6
210 DATA 47,16,254,211,254,
    62,13,6,55,16
220 DATA 254,211,254,1,15,
    59,8,111,195,236
230 DATA 234,122,179,40,12,
    221,110,0,124,173
240 DATA 103,62,1,55,195,10,
    235,108,24,244
250 DATA 121,203,120,16,254,
    48,4,6,66,16
260 DATA 254,211,254,6,62,
    32,239,5,175,60
270 DATA 203,21,194,249,234,
    27,221,35,6,49
280 DATA 62,127,219,254,31,
    208,122,60,194,
    227
290 DATA 234,6,59,16,254,
    201
```

```
EA6Ø              1Ø          ORG   6ØØØØ
EA6Ø              2Ø          ENT   6ØØØØ
EA6Ø  2AA5EA      3Ø          LD    HL,(VALOR+13)
EA63  DD2198EA    4Ø          LD    IX,VALOR
EA67  E5          5Ø          PUSH  HL
EA68  3EFD        6Ø          LD    A,#FD
EA6A  CDØ116      7Ø          CALL  #16Ø1
EA6D  AF          8Ø          XOR   A
EA6E  11A1Ø9      9Ø          LD    DE,#Ø9A1
EA71  CDØAØC      1ØØ         CALL  #ØCØA
EA74  FDCBØ2EE    11Ø         SET   5,(IY+2)
EA78  CDD415      12Ø         CALL  #15D4
EA7B  DDE5        13Ø         PUSH  IX
EA7D  1111ØØ      14Ø         LD    DE,#ØØ11
EA8Ø  AF          15Ø         XOR   A
EA81  CDA7EA      16Ø         CALL  GRABAR
EA84  DDE1        17Ø         POP   IX
EA86  Ø632        18Ø         LD    B,#32
EA88  76          19Ø LOOPØ   HALT
EA89  1ØFD        2ØØ         DJNZ  LOOPØ
EA8B  DD5EØB      21Ø         LD    E,(IX+11)
EA8E  DD56ØC      22Ø         LD    D,(IX+12)
EA91  3EFF        23Ø         LD    A,#FF
EA93  DDE1        24Ø         POP   IX
EA95  C3A7EA      25Ø         JP    GRABAR
EA98  Ø3          26Ø VALOR   DEFB  3
EA99  5Ø414E54    27Ø         DEFM  "PANTALLA"
EAA3  ØØ1BØØ4Ø    28Ø         DEFB  Ø,27,Ø,64
EAA7  213FØ5      29Ø GRABAR  LD    HL,#Ø53F
EAAA  E5          3ØØ         PUSH  HL
EAAB  2181Ø1F     31Ø         LD    HL,#1F8Ø
EAAE  CB7F        32Ø         BIT   7,A
EABØ  28Ø3        33Ø         JR    Z,LOOP1
EAB2  2198ØC      34Ø         LD    HL,#ØC98
EAB5  Ø8          35Ø LOOP1   EX    AF,AF'
EAB6  13          36Ø         INC   DE
EAB7  DD2B        37Ø         DEC   IX
EAB9  F3          38Ø         DI
EABA  3EØ3        39Ø         LD    A,#Ø3
EABC  47          4ØØ         LD    B,A
EABD  1ØFE        41Ø LOOP2   DJNZ  LOOP2
EABF  D3FE        42Ø         OUT   (#FE),A
EAC1  EEØF        43Ø         XOR   #ØF
EAC3  Ø6A4        44Ø         LD    B,#A4
EAC5  2D          45Ø         DEC   L
EAC6  2ØF5        46Ø         JR    NZ,LOOP2
EAC8  Ø5          47Ø         DEC   B
EAC9  25          48Ø         DEC   H
EACA  F2BDEA      49Ø         JP    P,LOOP2
EACD  Ø62F        5ØØ         LD    B,#2F
```

```
EACF  1ØFE    51Ø  LOOP3   DJNZ  LOOP3
EAD1  D3FE    52Ø           OUT   (#FE),A
EAD3  3EØD    53Ø           LD    A,#ØD
EAD5  Ø637    54Ø           LD    B,#37
EAD7  1ØFE    55Ø  LOOP4   DJNZ  LOOP4
EAD9  D3FE    56Ø           OUT   (#FE),A
EADB  Ø1ØF3B  57Ø           LD    BC,#3BØF
EADE  Ø8      58Ø           EX    AF,AF'
EADF  6F      59Ø           LD    L,A
EAEØ  C3ECEA  6ØØ           JP    LOOP7
EAE3  7A      61Ø  LOOP5   LD    A,D
EAE4  B3      62Ø           OR    E
EAE5  28ØC    63Ø           JR    Z,LOOP8
EAE7  DD6EØØ  64Ø           LD    L,(IX+Ø)
EAEA  7C      65Ø  LOOP6   LD    A,H
EAEB  AD      66Ø           XOR   L
EAEC  67      67Ø  LOOP7   LD    H,A
EAED  3EØ1    68Ø           LD    A,#Ø1
EAEF  37      69Ø           SCF
EAFØ  C3ØAEB  7ØØ           JP    LOOP13
EAF3  6C      71Ø  LOOP8   LD    L,H
EAF4  18F4    72Ø           JR    LOOP6
EAF6  79      73Ø  LOOP9   LD    A,C
EAF7  CB78    74Ø           BIT   7,B
EAF9  1ØFE    75Ø  LOOP1Ø  DJNZ  LOOP1Ø
EAFB  3ØØ4    76Ø           JR    NC, LOOP12
EAFD  Ø642    77Ø           LD    B,#42
EAFF  1ØFE    78Ø  LOOP11  DJNZ  LOOP11
EBØ1  D3FE    79Ø  LOOP12  OUT   (#FE),A
EBØ3  Ø63E    8ØØ           LD    B,#3E
EBØ5  2ØEF    81Ø           JR    NZ,LOOP9
EBØ7  Ø5      82Ø           DEC   B
EBØ8  AF      83Ø           XOR   A
EBØ9  3C      84Ø           INC   A
EBØA  CB15    85Ø  LOOP13  RL    L
EBØC  C2F9EA  86Ø           JP    NZ,LOOP1Ø
EBØF  1B      87Ø           DEC   DE
EB1Ø  DD23    88Ø           INC   IX
EB12  Ø631    89Ø           LD    B,#31
EB14  3E7F    9ØØ           LD    A,#7F
EB16  DBFE    91Ø           IN    A,(#FE)
EB18  1F      92Ø           RRA
EB19  DØ      93Ø           RET   NC
EB1A  7A      94Ø           LD    A,D
EB1B  3C      95Ø           INC   A
EB1C  C2E3EA  96Ø           JP    NZ,LOOP5
EB1F  Ø63B    97Ø           LD    B,#3B
EB21  1ØFE    98Ø  LOOP14  DJNZ  LOOP14
EB23  C9      99Ø           RET
```

MAPA, CARGADOR Y POKES PARA...

FUTURE KNIGHT



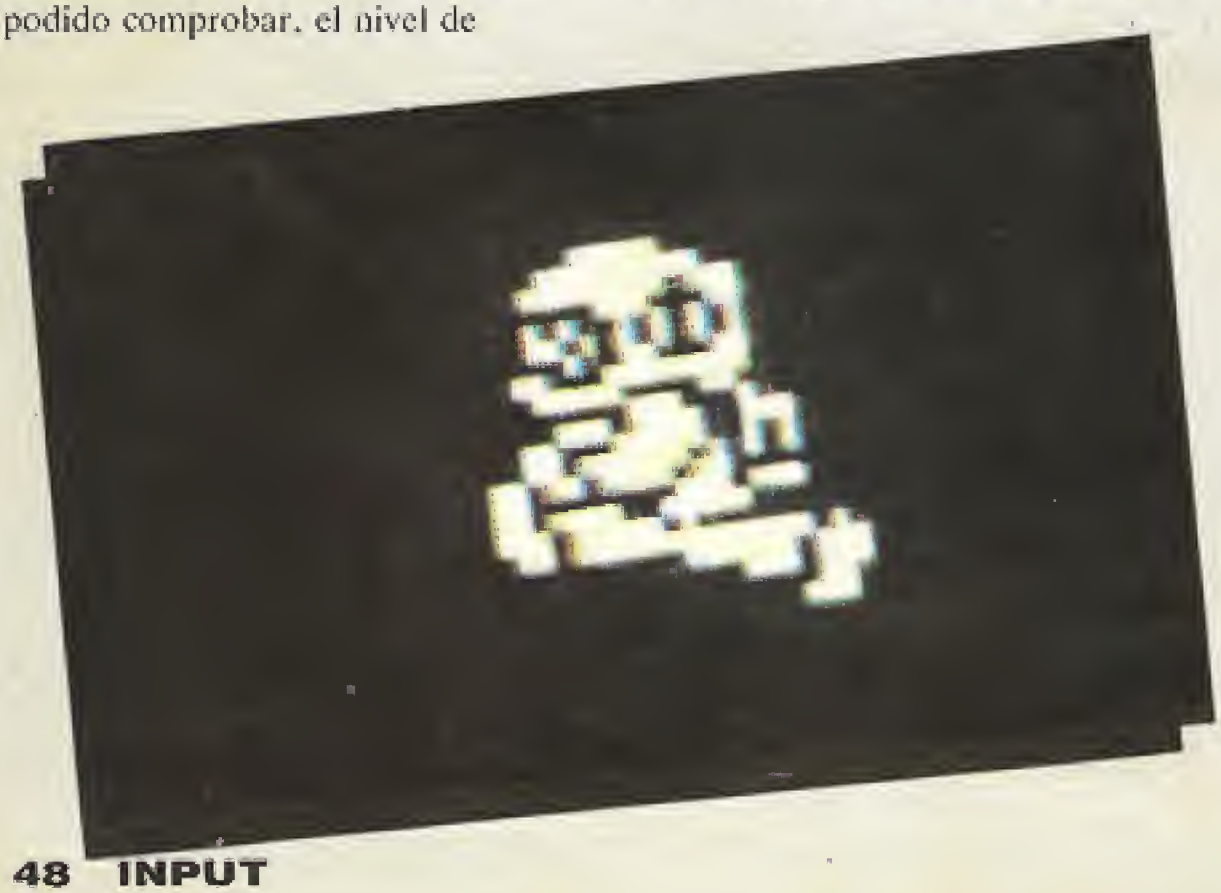
En un lugar del Espacio Interestelar se halla la nave SSROBUSTEC, de la gran flota imperial. En ella se recibe un mensaje inquietante y todos los oficiales de telecomunicaciones se dirigen a sus puestos. Tras denodados esfuerzos consiguen descifrarlo. «Soy la princesa STALINA, me ha raptado un ser malvado, medio hombre medio máquina, sólo puedo decir que para llegar hasta mí hay que atravesar 20 niveles y luchar con un robot casi invencible, GENCHODROID. Me encuentro en...» Aquí la comunicación se interrumpió.

SIR RADOLF, sin dudarlo un instante, salió disparado hacia sus aposentos, de los que regresó investido con una reluctante y preciosa armadura. «El deber me llama», dijo SIR RADOLF, quien ordenó a sus compañeros que se dirigieran inmediatamente al planeta. Sus compañeros y demás tripulantes se despidieron de él como si fuese la última vez que lo fuesen a ver.

SIR RADOLF caminó hacia la salida y parafraseó la histórica frase «ALEA IACTA EST».

Como muchos de vosotros habréis podido comprobar, el nivel de dificultad del FUTURE KNIGHT es muy elevado. Para que podáis llegar al final, INPUT os proporciona el mapa completo, con sus veinte niveles. En este número se publican los diez primeros, mientras que los diez restantes aparecerán en el próximo. Y no sólo el mapa sino también el ansiado CARGADOR para que disfrutéis de ENERGIA Infinita.

Para utilizar el CARGADOR y, por tanto, ENERGIA Infinita, deberéis teclearlo y acto seguido salvarlo. Una vez grabado hay que ejecutarlo e introducir el original en el cassette, pulsar play y dejar que el programa se cargue hasta el final.



CONSEJOS PARA SOBREVIVIR

Si queréis coronar con éxito vuestra aventura debéis seguir atentamente estos consejos.

Partimos del primer nivel en el que, para comenzar, cogeremos un objeto llamado «SAFE PASS» con el que nos dirigiremos hacia la salida (Exit). Ahora ya estaremos en el nivel dos.

En este nivel hay tres salidas, deberemos dirigirnos a la superior para pasar al nivel tres. Nos encontramos aquí con un objeto llamado «CONFUSER». No deberemos apropiárnoslo puesto que perderíamos el «SAFE PASS» que habíamos conseguido antes.

Acto seguido nos dirigiremos al final de dicho nivel para salir y pasar al 4.

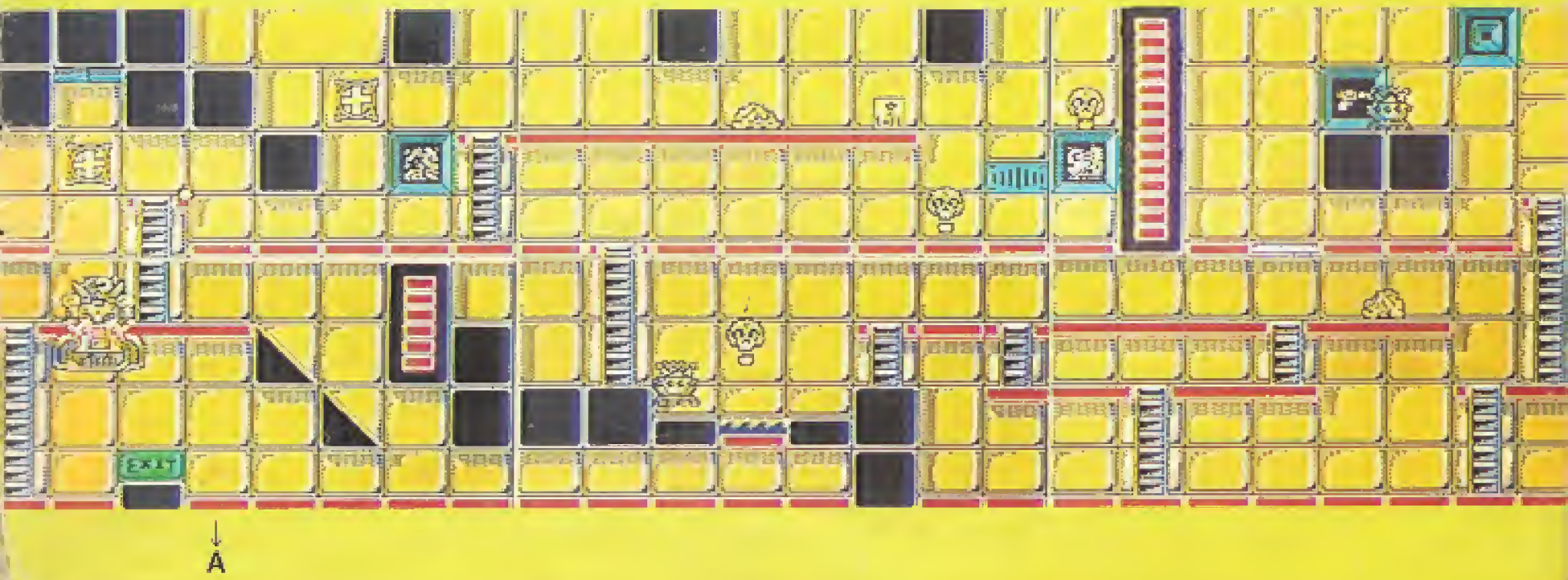
En éste deberéis cambiar el «SAFE PASS» por la «PLATFORM KEY».

Acto seguido nos dirigiremos a la salida superior de este nivel para pasar al 5.

Una vez en él deberemos avanzar siempre hacia la derecha, donde cambiaremos el objeto que llevamos por la «SECURO KEY».
Volveremos sobre nuestros pasos hacia el nivel 4. Nos dirigiremos hacia la salida que hay abajo, pero a la izquierda.
Nos situaremos encima de ella y presionaremos la tecla «U», con lo que se abrirá. Pero antes de entrar al nivel 6 cogeremos de nuevo el «SAFE PASS».
Ya en el nivel 6, avanzaremos hacia abajo hasta la salida y otra vez encima pulsaremos la tecla «U» con lo que se abrirá, pasando al nivel 7.
Volveremos a ir hacia abajo para coger el llamado «EXIT PASS». Una vez lo tengamos en nuestro poder retrocederemos hasta el nivel 2, desde el que nos dirigiremos a la salida que hay abajo a la izquierda y pasaremos al nivel 8.
En este nivel sólo hay una salida, sobre la que tendremos que situarnos y pulsar la tecla «U», pasando al nivel 9. Nos dirigiremos a la única salida que hay y entraremos, por fin, en el nivel 10. Entonces comunicaremos hacia el final de este nivel y cogeremos el objeto llamado «RELEASE SPELL». Con él en nuestras manos y embargados por la emoción nos situaremos encima de la salida y...

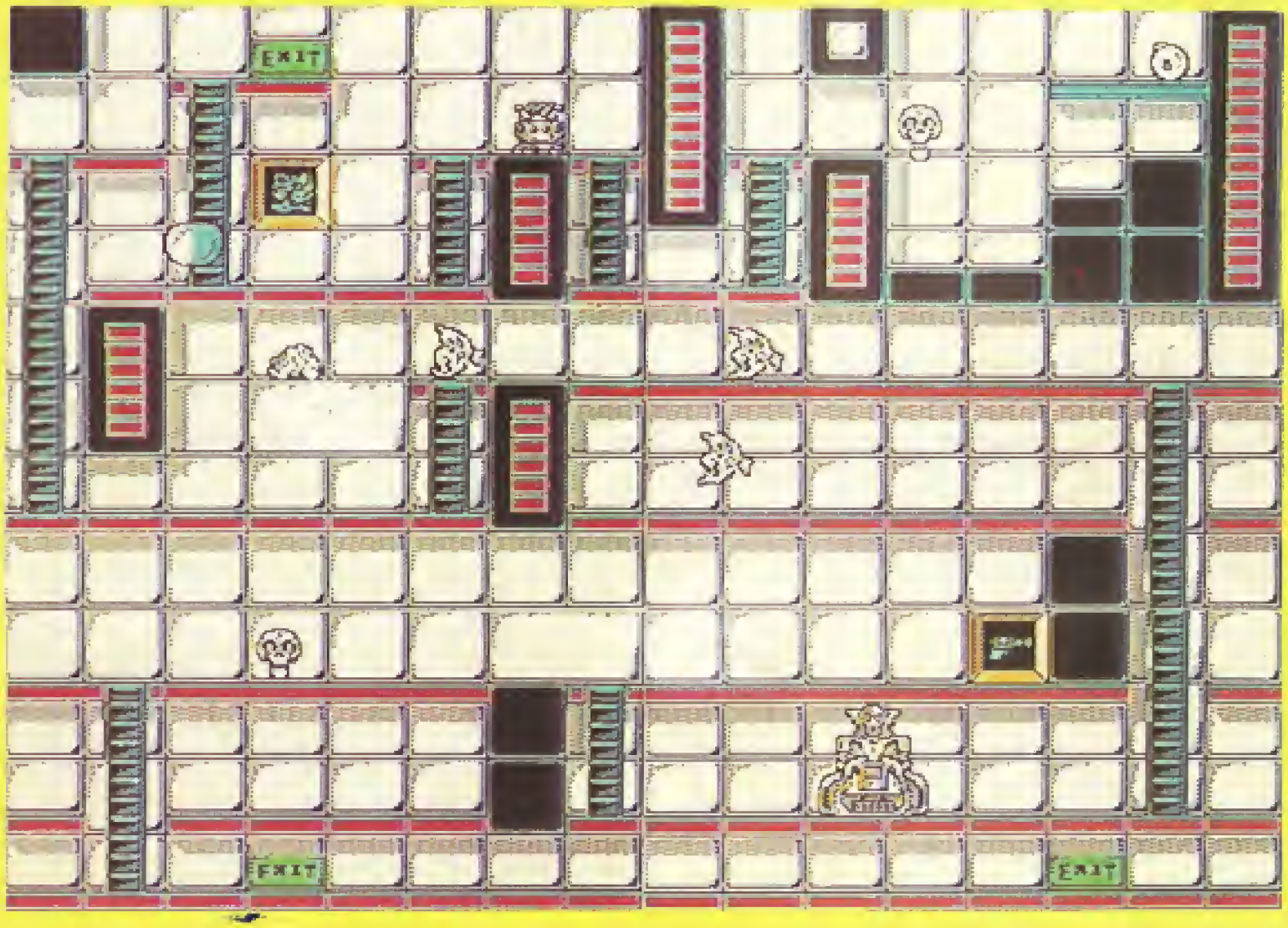
```
 5 REM CARGADOR PARA
   INPUT POR
 7 REM         JOSE VILA
 8 REM  ***********************
 9 GO TO 6Ø
1Ø BORDER Ø: PAPER Ø: INK 7:
   BRIGHT 1: CLS
2Ø PRINT AT 8,5;"INTRODUCE
   EL ORIGINAL"
3Ø PRINT INK 4;AT 1Ø,Ø;"DEL
   FUTURE KNIGHT Y PULSA
   'PLAY'"
35 PRINT FLASH 1;AT 15,9;
   "PULSA UNA TECLA"
4Ø PAUSE Ø: CLS: INK Ø:
   RANDOMIZE USR 5ØØØØ
6Ø FOR N=5ØØØØ TO 5Ø119:
   READ A: POKE N,A: NEXT N
7Ø GO TO 1Ø
1ØØ DATA 221,33,Ø,Ø,17,17,Ø,
    62,Ø,55,2Ø5,86,5,221,33,
    Ø,Ø,17,12Ø,4
1Ø1 DATA 62,255,55,2Ø5,86,5,
    221,33,Ø,Ø,17,17,Ø,62,Ø,
    55,2Ø5,86,5,221
1Ø2 DATA 33,Ø,Ø,17,62,Ø,62,
    255,55,2Ø5,86,5,221,33,
    Ø,Ø,17,17,Ø,62
1Ø3 DATA Ø,55,2Ø5,86,5,221,
    33,Ø,128,17,188,2,62,
    255,55,2Ø5,86,5,33,125
1Ø4 DATA 13Ø,34,5,129,62,33,
    5Ø,125,13Ø,33,Ø,Ø,34,
    126,13Ø,62,34,5Ø,128,13Ø
1Ø5 DATA 33,9,187,34,129,
    13Ø,62,195,5Ø,131,13Ø,
    33,Ø,68,34,132,13Ø,195,
    Ø,128
```

MAPA DEL FUTURE KNIGHT



B

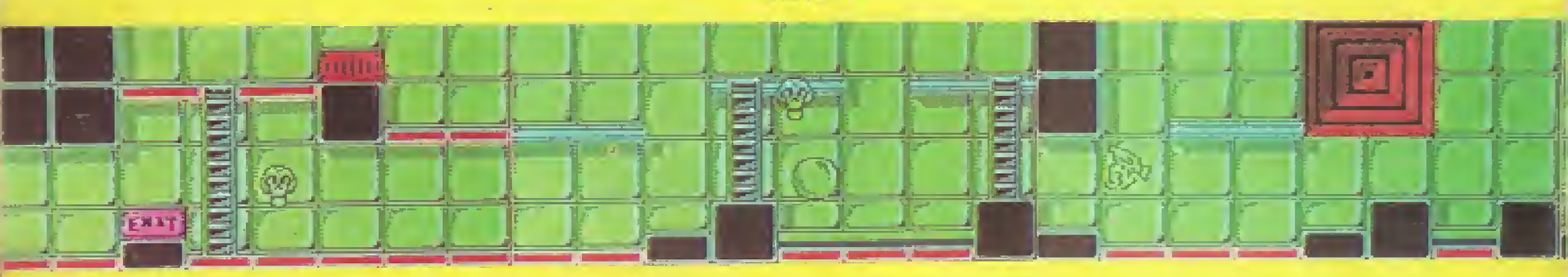
NIVEL 2



G

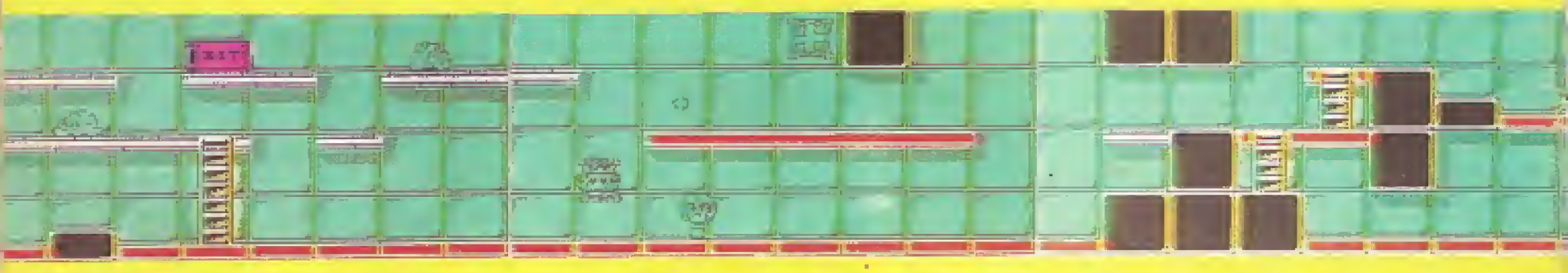
A

NIVEL 3



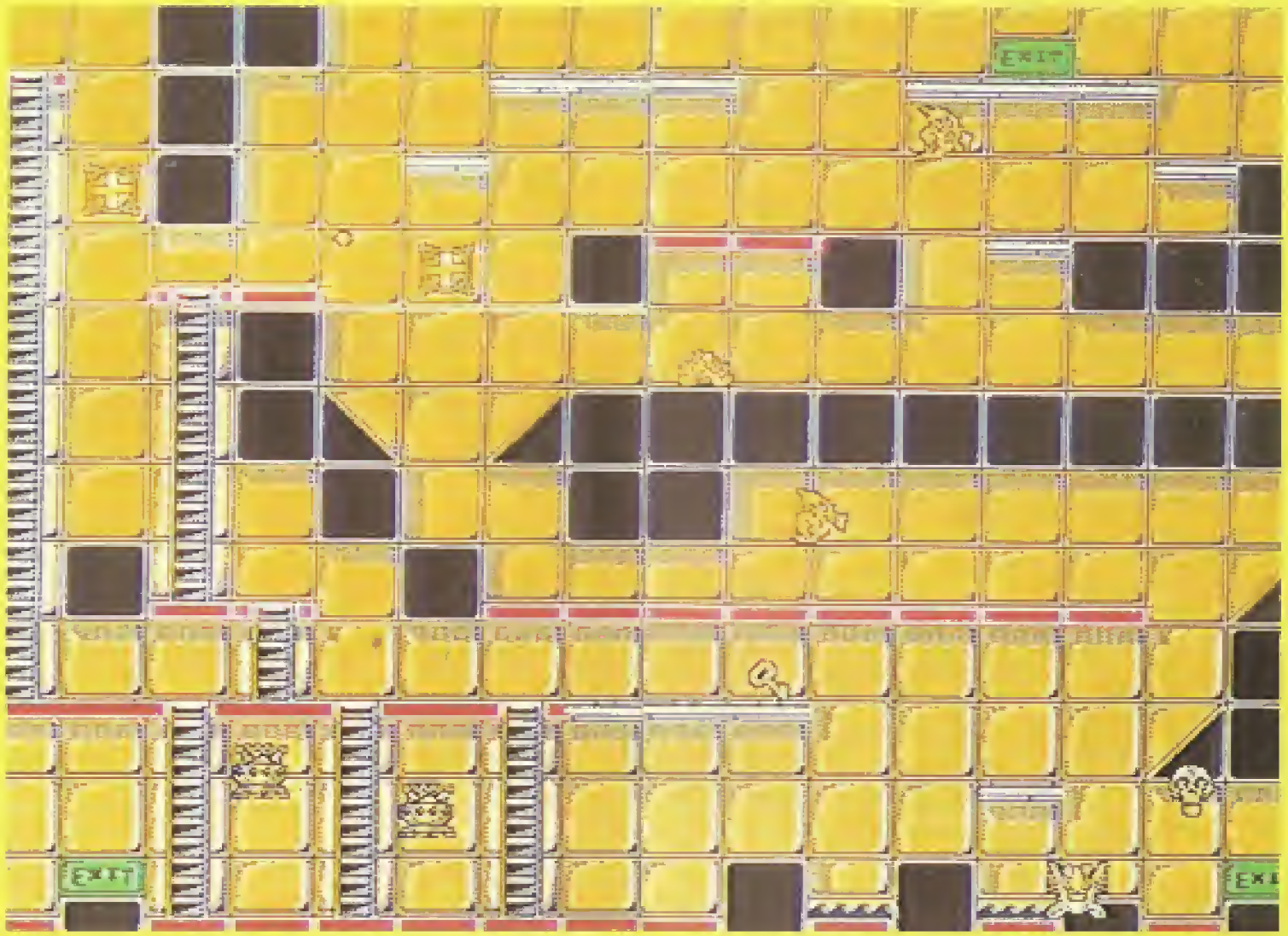
C

NIVEL 5



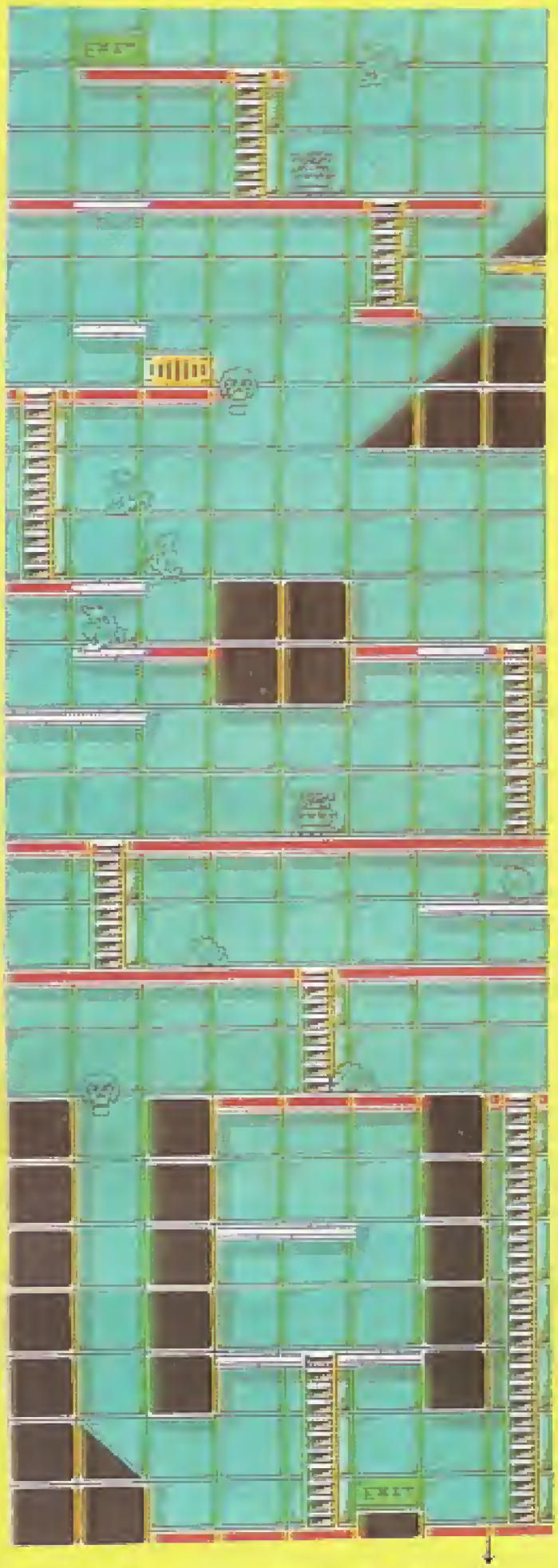
D

NIVEL 4



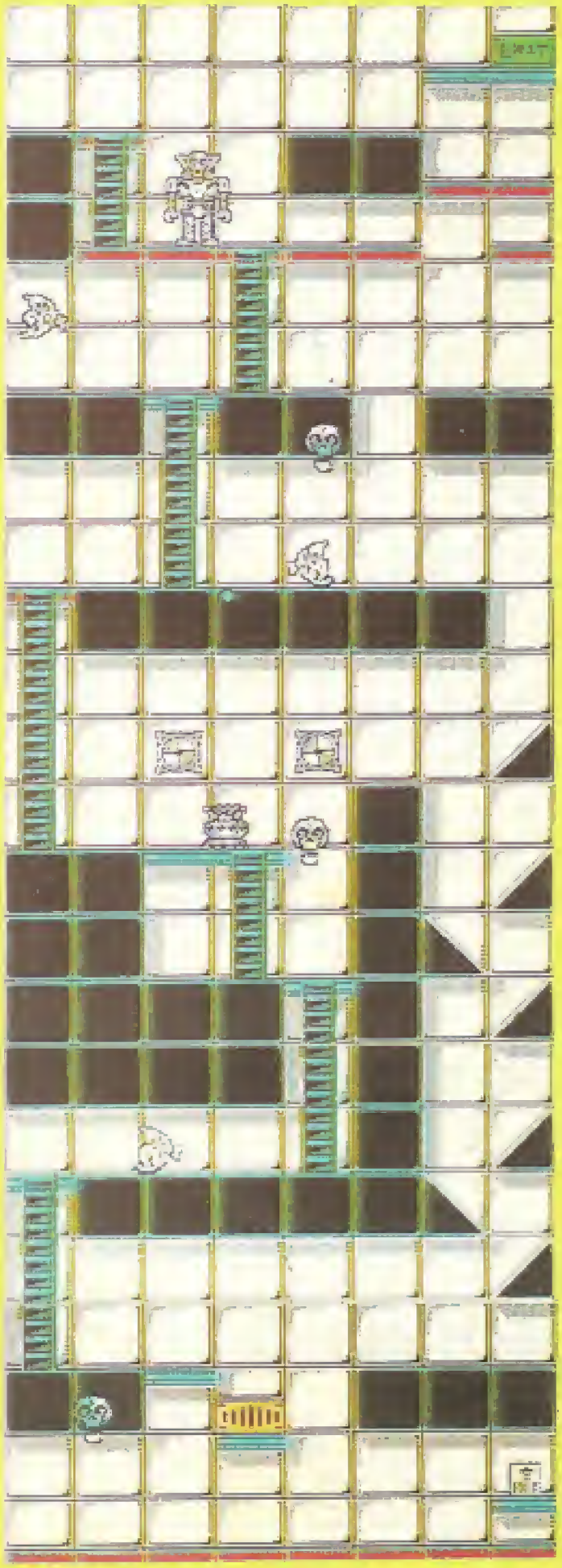
E

NIVEL 6



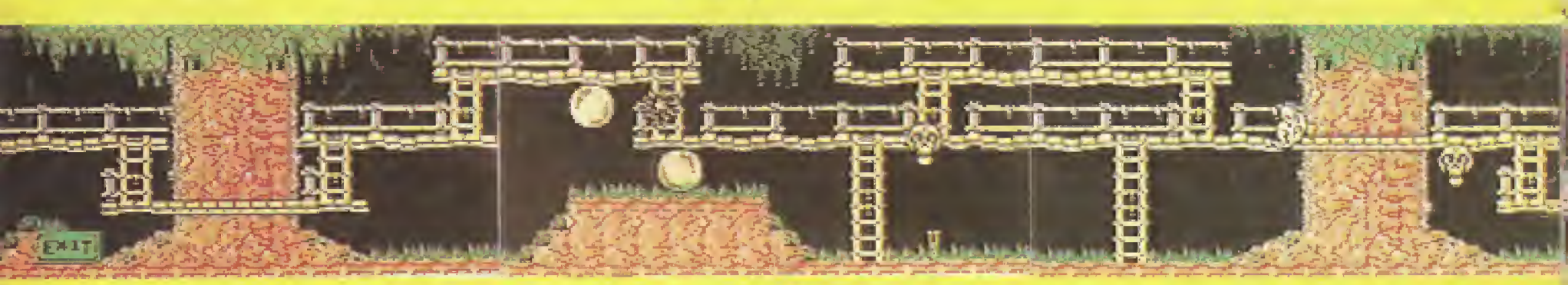
F

NIVEL 7



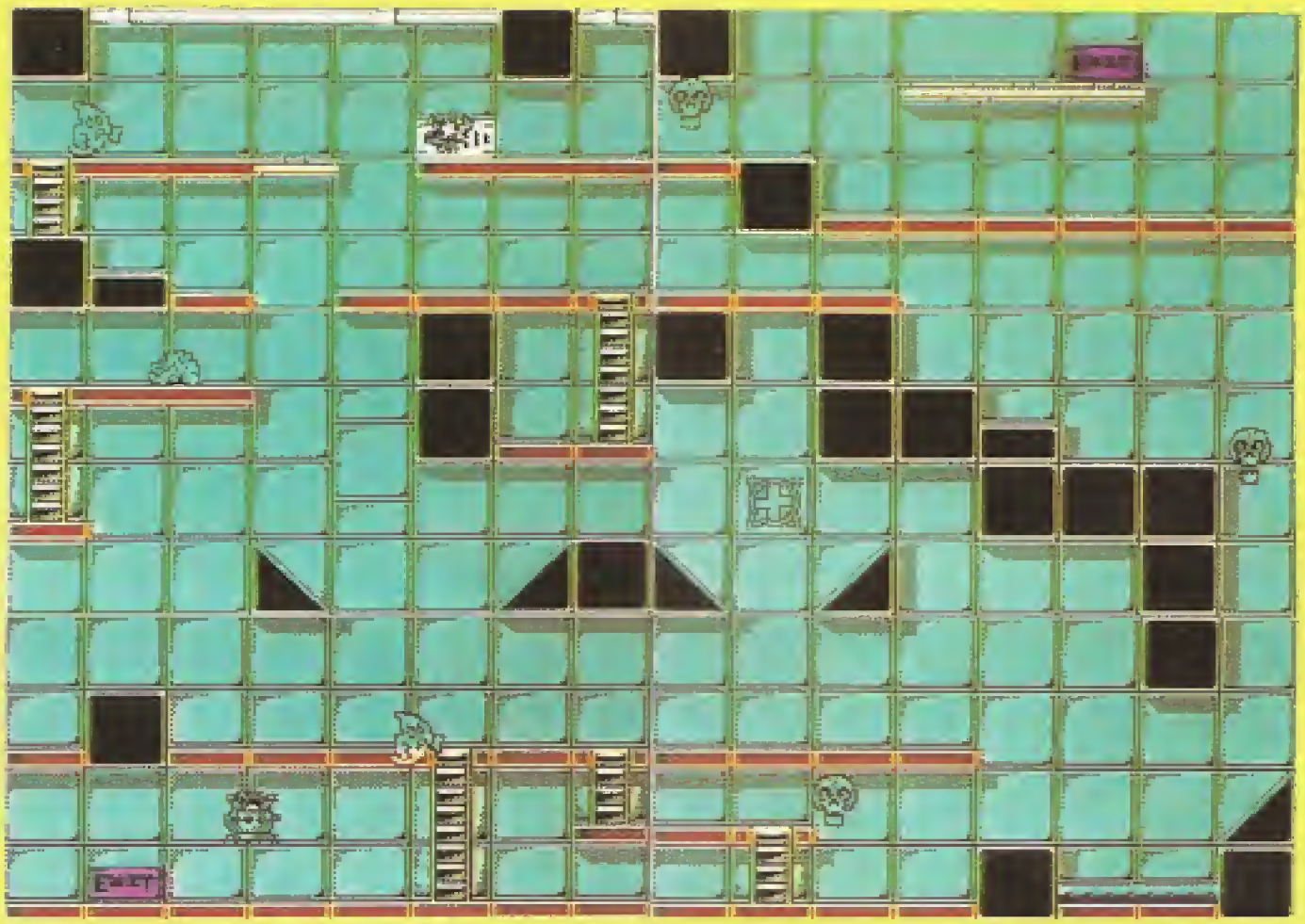
I

NIVEL 10



NIVEL 8

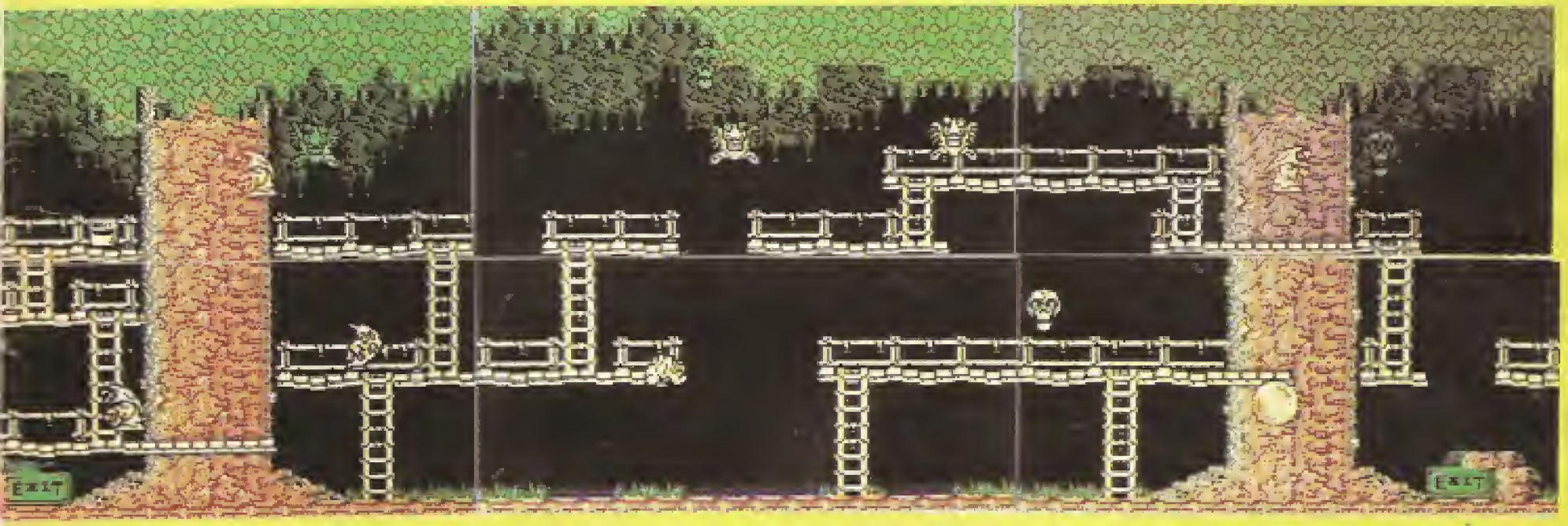
G ↑



↓ H

H ↑

NIVEL 9



↓ SEGUNDA PARTE

SOFTACTUALIDAD

HYPERBOWL

Este programa puede considerarse como una curiosa mezcla entre simulador deportivo, arcade, y aventura futurista. Consiste exactamente en reproducir el deporte favorito del público del siglo XXXVI, una especie de hockey sobre el vacío, practicado en el espacio abierto con sofisticadas naves. Puedes elegir nivel de dificultad, número de jugadores, e incluso estudiar un catálogo de naves disponibles para escoger la que más te guste. HYPERBOWL es un juego muy adictivo y, además, muy original.

NINJA

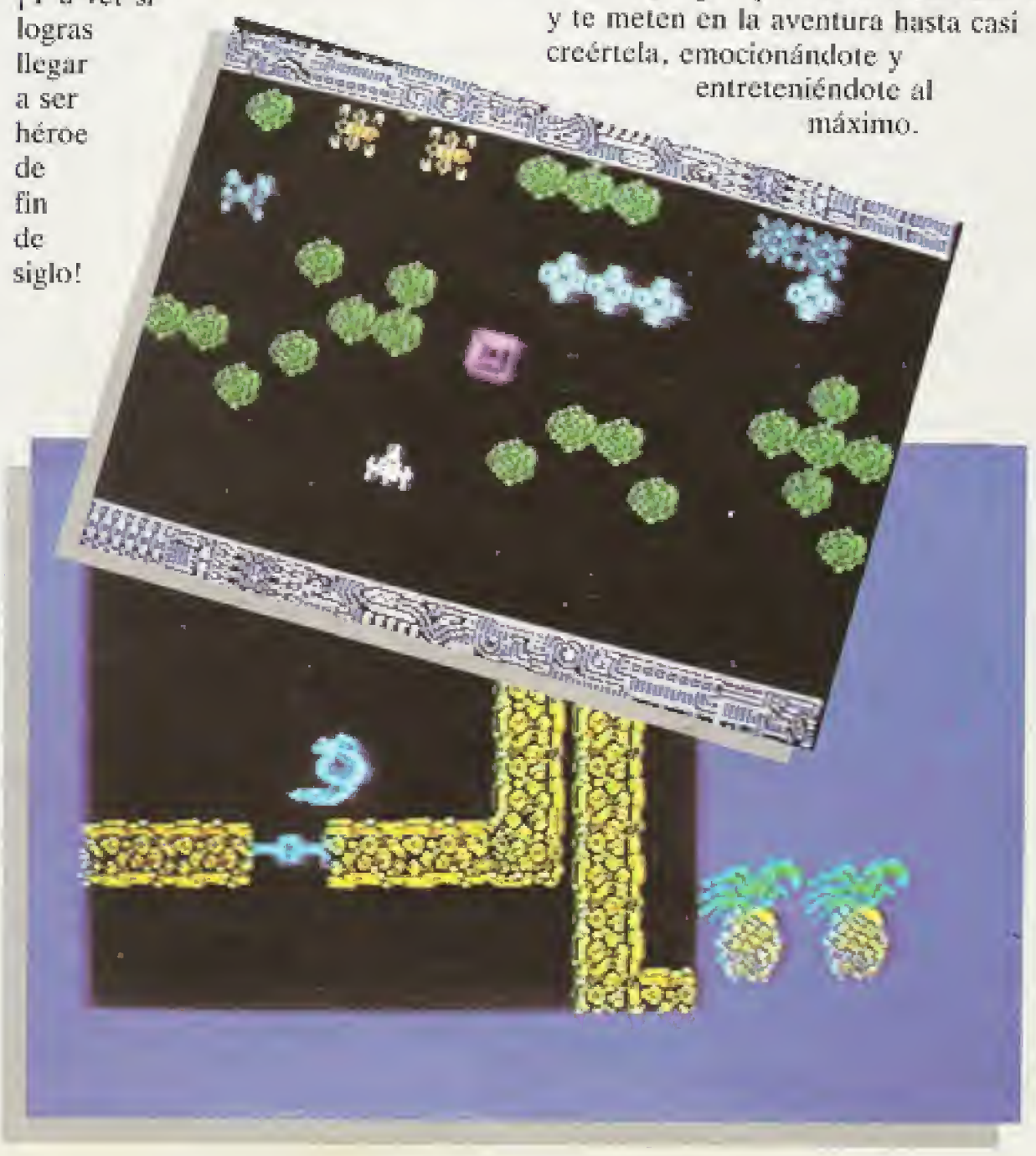
Ninja es un nuevo programa de artes marciales, planteado al estilo del FIST II, pero con una resolución gráfica más modesta. Los golpes y fintas son poco numerosos y poco espectaculares, pero el desarrollo general de la acción está muy bien pensado, y el interés aumenta a medida que se juega. Por ello, podemos considerar a este programa como una buena vídeo-aventura y un mediocre simulador.

HOW TO BE A HERO

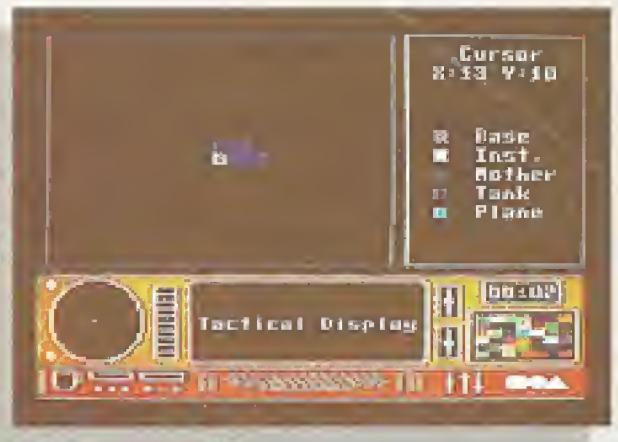
¿Quieres saber cómo convertirte en un héroe? Seguro que un arcade como «HOW TO BE A HERO» (Cómo ser un héroe) responderá sobradamente a tu pregunta. Realmente, puedes considerarte un héroe si llegas hasta el final. Con tres escenarios completamente diferentes, misiones diversas, y una enorme cantidad de mortíferos enemigos. Prepárate a perder vidas!! ¡Y a ver si logras llegar a ser héroe de fin de siglo!

XCEL

XCEL es un arcade tradicional, aderezado con un buen número de efectos gráficos de gran originalidad, y computador que «traduce» textos secretos escritos en lengua alienígena. Sin duda, se trata de uno de esos juegos que «hacen ambiente» y te meten en la aventura hasta casi creértela, emocionándote y entreteniéndote al máximo.



SIMULADORES DE VUELO



Hasta finales del verano pasado, el cajón más desocupado de nuestros archivos era precisamente el dedicado a los simuladores. Apenas había en él una docena de fichas, y todo parecía indicar que seguiría así durante mucho tiempo. Sin embargo, cuando la mayoría de los «incondicionales» del género ya empezábamos a perder las esperanzas, el panorama cambió repentinamente. En pocos meses, la lista de programas de simulación se fue engrosando rápidamente con títulos de extraordinaria calidad, desbordando las más optimistas expectativas. Aunque el éxito comercial de estos programas no fue tan grande como se esperaba en muchos casos, los resultados fueron lo suficientemente positivos como para convencer a los fabricantes de la rentabilidad de la fórmula. Así pues, desde entonces tenemos oportunidad de comentaros el lanzamiento de al menos un programa de estas características cada mes, cuando no son dos o hasta tres, siempre con las más favorables calificaciones.

Estamos al corriente del interés que muestran muchos de nuestros lectores por el tema, que en los últimos tiempos ha ganado una enorme cantidad de adeptos, y por ello hemos decidido elaborar un informe sobre las novedades disponibles, incluyendo una breve recapitulación sobre los «clásicos», y un comentario bastante detallado de los títulos más recientes. En este número, os ofrecemos un monográfico dedicado exclusivamente a los simuladores de vuelo. En el futuro, publicaremos otro similar referido a los juegos de simulación deportiva.

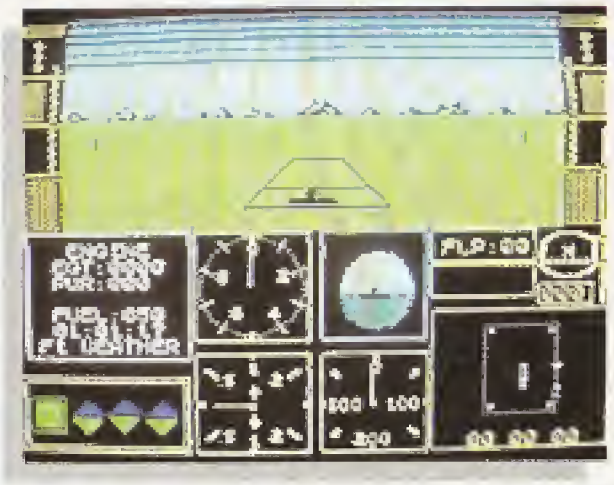
LOS COMIENZOS

A la par que los primeros Spectrum de 16 K, llegaron a España varios simuladores de vuelo con diversos nombres y un mismo contenido: el objetivo del juego consistía simplemente en despegar, volar hasta un lugar determinado, y aterrizar de nuevo. Como todo ello no solía realizarse en menos de veinte o treinta minutos, y además había una opción de piloto automático, estos programas resultaban tremendamente aburridos, aunque a falta de otra cosa, hasta los más exigentes se conformaban. Poco después aparecieron dos títulos que mantuvieron su hegemonía durante varios años (algo insólito en este mundillo del soft), y que todavía hoy pueden competir con los mejores. Estamos hablando de FIGHTER PILOT Y COMBAT LYNX (disponibles en tiendas especializadas, aunque a veces puede resultar difícil encontrarlos, sobre todo el segundo). Ambos programas incluían el aliciente del combate, la estrategia y la acción, además de desarrollar nuevas y sofisticadas técnicas que sentaron el precedente para posteriores mejoras.

Más tarde se dio un salto cualitativo importante, con la aparición de nuevos programas en los que se buscaba una orientación más comercial, restando sofisticación a su manejo, sin perder por ello realismo. En esta línea destacaron —y siguen destacando— DAM BUSTERS y SKYFOX, ambos recomendables no sólo a los apasionados del tema, sino también a todos aquellos que quieran probar emociones fuertes.

LO MAS RECIENTE

Durante el verano pasado, se publicaron en Inglaterra ACE y STRIKE FORCE HARRIER, dos sensacionales programas en los que se reúnen todos los aciertos de los simuladores precedentes. Los dos permiten realizar las más complejas



tácticas y maniobras del combate aéreo con un sorprendente grado de realismo, apoyándose en un soporte gráfico difícilmente superable.
Desde la publicación de estos dos programas, aún no se ha conseguido mejorar el nivel alcanzado, aunque constantemente aparecen nuevos simuladores de una calidad que se puede considerar al menos equiparable.

NOVEDADES

Los últimos simuladores de vuelo publicados hasta el momento de redactar este informe, son ACROJET, realizado por la firma Micro-Prose junto con un estupendo simulador de combate naval (SILENT SERVICE); DEEP STRIKE, un original programa que reproduce los combates aéreos que tuvieron lugar en la Primera Guerra Mundial; ACE OF ACES, comentado con detalle más adelante; y TOP GUN.
Aparte de estos títulos, pronto estará disponible para Spectrum la última superproducción de Micro-Prose, cuyo desarrollo ha costado cerca del *millón y medio de dólares*. Se trata de GUN SHIP, un simulador de vuelo en helicóptero con opciones de combate que promete constituirse, según todos los indicios, en el mayor bombazo del año.

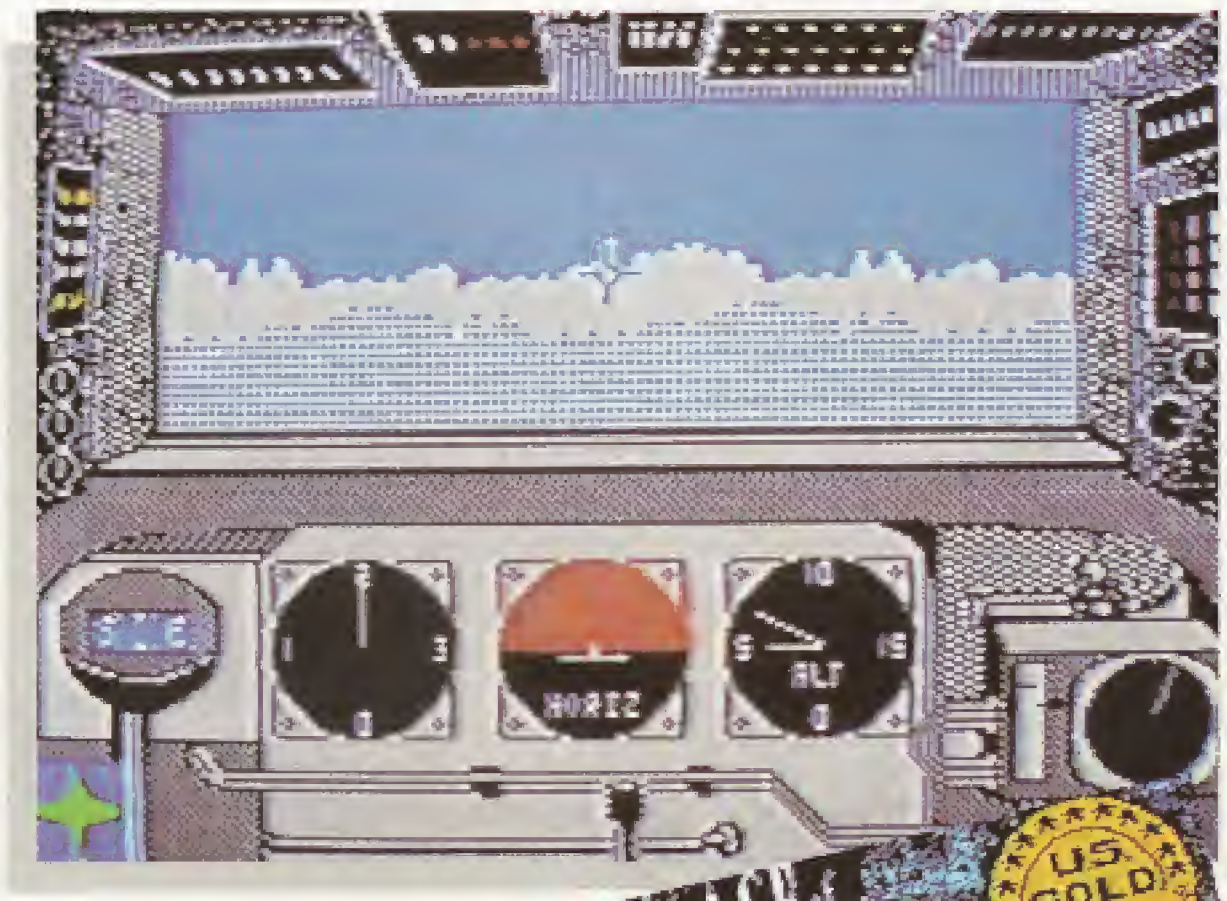
«OTROS» SIMULADORES

Existen también numerosos programas que son presentados como «simuladores», cuando en realidad serían más clasificables como «arcades», teniendo siempre muy en cuenta que un verdadero simulador debe ser, ante todo, fiel a la realidad.
Dos ejemplos de buenos programas arcade que incorporan algunos rasgos propios de los simuladores son los de INFILTRATOR, publicado durante las pasadas navidades, y FRACTALUS. Este último se ambienta en un planeta desconocido cuyos paisajes son generados aleatoriamente por el ordenador a través de «fractales».
Más recientemente se ha publicado en nuestro país *1942*, un arcade cuajado de acción que difícilmente podría clasificarse como simulador, pero que tampoco podríamos dejar de reseñar.

TITULO	CALIFICACION		
	REALISMO	GRAFICOS	INTERES
COMBAT LYNX	*7*	*8*	*8*
FIGHTER PILOT	*8*	*8*	*8*
DAM-BUSTERS	*8*	*9*	*10*
SKYFOX	*8*	*9*	*9*
ACE	*9*	*9*	*9*
STRIKE FORCE HARRIER	*9*	*10*	*9*
ACROJET	*8*	*8*	*8*
TOP GUN	*7*	*7*	*9*
DEEP STRIKE	*8*	*8*	*8*
ACE OF ACES	*9*	*9*	*9*

ACE OF ACES

● US GOLD ■ SIMULADOR



¿Recordáis el fabuloso programa THE DAM BUSTERS, que llegó a situarse en los más altos niveles de ventas del mercado de COMMODORE y que gracias a esto han ido saliendo nuevas versiones para posteriores ordenadores, como es el caso del MSX? Pues ahora la prestigiosa firma U.S.GOLD nos presenta su nuevo simulador de vuelo ACE OF ACES que bien seguro sabrán apreciar los adictos a este tipo de programas; y llegará a batir todos los récords de permanencia en las listas de popularidad conseguidos por su antecesor. Con este nuevo programa podrás recrear las arriesgadas misiones que protagonizaron los míticos cazabombarderos «Mosquito» durante la Segunda Guerra Mundial, además de disfrutar con un original replanteamiento de los esquemas tradicionales en este tipo de juegos.
Su detalle más destacable es la gran diversidad de opciones que ofrece, almacenables en memoria a través de un sistema de «carga múltiple», muy eficaz pero algo incómodo.
Después de hacer una selección previa en el programa cargador, podrás acceder a un variado número de misiones diferentes:
Combatir contra los cazas enemigos, interceptar bombas V-1, atacar convoys de submarinos, bombardear líneas férreas y trenes de prisioneros, o simplemente practicar para lograr una mayor destreza.
El avión se controla a través de cinco pantallas: vista del navegador, vista del bombardero, y vistas al frente y *a los lados* (atención a este novedoso detalle) del piloto. Las vistas a los lados permiten no sólo controlar los mandos colocados en ellas, sino también observar directamente el estado de los motores, o seguir visualmente algún avión enemigo.
ACE OF ACES es uno de los mejores simuladores de vuelo de combate disponibles por el momento y una de las claves de su éxito es que ha sabido sintetizar las mejores cualidades de los programas anteriores a él y de este modo lograr un producto de muy alta calidad tanto en lo referido a los gráficos y movimiento como a la trama y nivel de dificultad del juego. El problema antes citado de la «carga mútliple» es en el fondo una solución y ventaja ya que nos permite disponer de diferentes misiones gracias a un programa común a todas ellas y esto amplía y diversifica las posibilidades de este completo programa. No te lo pierdas.

ANIMACION	9
INTERES	9
GRAFICOS	9
COLOR	8
SONIDO	7
TOTAL	42

LAST MISSION

● OPERASOFT ■ ARCADE

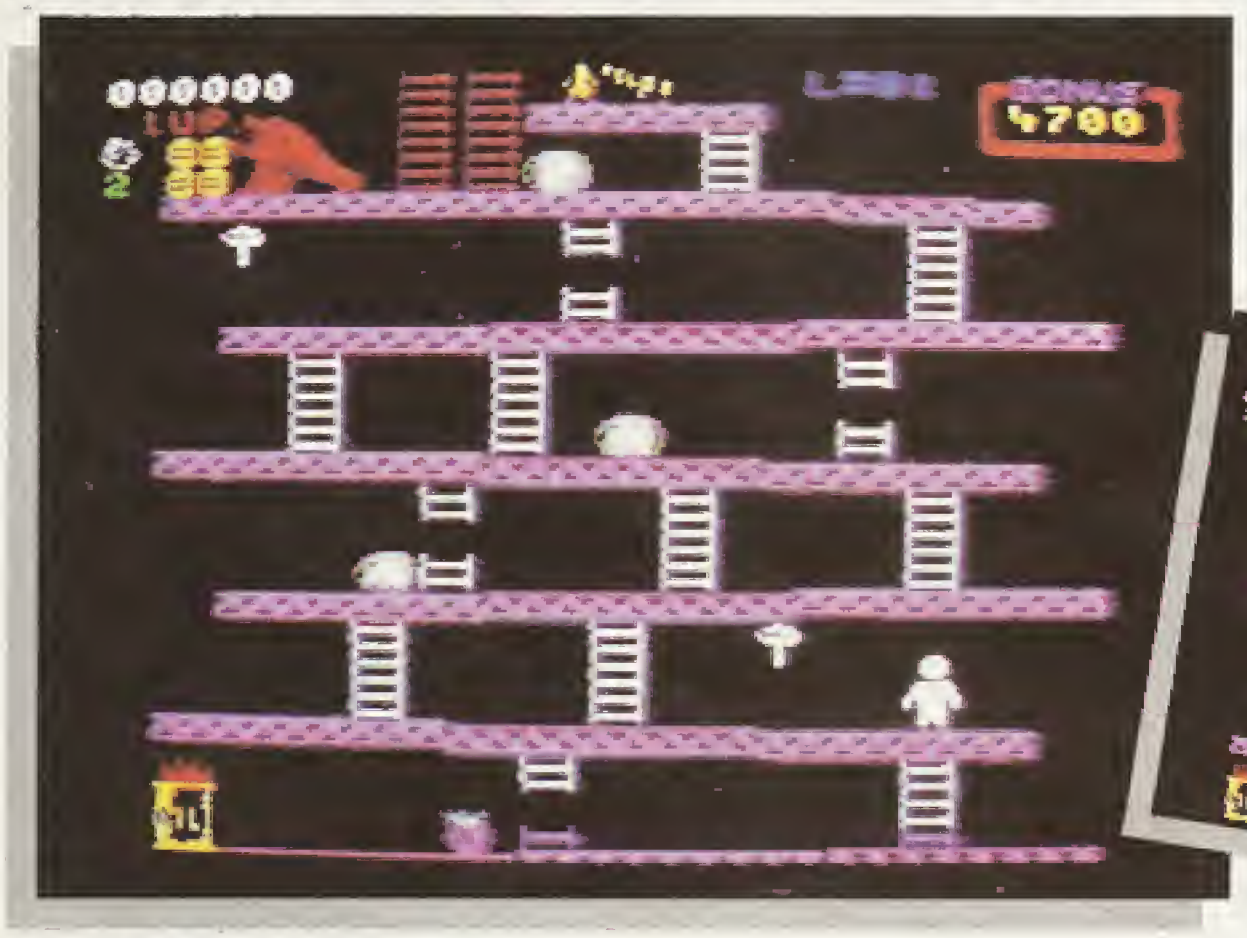
Según nuestros informes, la versión para SPECTRUM de LAST MISSION no saldrá a la venta hasta el mes de mayo aproximadamente (es decir, de forma simultánea a la publicación de este número), aunque, como algunos ya sabréis, su versión AMSTRAD está en el mercado desde marzo.
El protagonista del programa es un curioso artefacto mecánico con forma de vehículo-oruga, capaz de desprender una parte de su estructura y hacerla volar a velocidades vertiginosas. Su misión consiste en escapar del interior de una base enemiga, enclavada en el subsuelo de un planeta desconocido, y volver a la Tierra. Para lograrlo será preciso atravesar varios niveles de defensas en creciente grado de dificultad, y después de haber superado un sinfín de mortíferos obstáculos, llegar hasta la nave que aguarda en la superficie.
En definitiva, se trata de un arcade altamente adictivo, con un nivel de dificultad elevado, y una calidad técnica sobresaliente. El único aspecto que no merece una valoración muy favorable es la originalidad, dado su ligero parecido con un magnífico programa de MIKRO-GEN publicado el año pasado, con el título de EQUINOX.
Esperamos impacientes que pronto esté disponible la versión SPECTRUM de LAST MISSION, programa que hace el número tres de los publicados por OPERA SOFT, y que probablemente supondrá la consagración del grupo.

DONKEY KONG

● OCEAN ■ RESCATAR A LA RUBIA

DONKEY KONG es, sin duda, uno de los «clásicos» más queridos por los viejos aficionados a los vídeo-juegos. Aunque hay quien afirma que está demasiado desfasado para los tiempos que corren, nosotros nos ponemos de parte de los nostálgicos y, sin entrar en más detalles, os lo recomendamos. No es un programa espectacular de grandes artificios gráficos, pero aún conserva ese «algo» que le convirtió hace unos años en el más popular vídeo-juego del mundo.
El argumento es sencillo y similar al de la película *King Kong*, de la cual existen, como recordarás, dos versiones, una de 1931 y la otra, realizada en el año 1976, en color y pantalla ancha. Un gorila enamoradizo pero fiero, se encariña de tu novia y decide raptarla. Tú, superando sus ansias posesivas deberás liberarla sorteando los barriles que te lanza. Cuentas con algunas ayudas adicionales para cumplir tu misión.

ANIMACION	6
INTERES	4
GRAFICOS	4
COLOR	4
SONIDO	5
TOTAL	23



SPELL BOUND-KNIGHT TYME

● MASTERTRONIC ■ AVENTURA DE ESTRATEGIA

Después del clamoroso éxito obtenido con FINDERS KEEPERS, el prolífico y creativo David Jones vuelve a la carga con dos nuevos programas planteados como continuaciones sucesivas de la aventura anterior. Se trata de SPELL BOUND y KNIGHT TYME.

En ambos juegos, el protagonista es el ya mítico MAGIC KNIGHT (el Caballero Mágico), que al igual que en las ocasiones precedentes, tendrá como misión sacar de algún apuro a su «tutor» el mago GIMBAL.

En líneas generales, el argumento coincide con el clásico modelo «coger, usar y dejar», sólo que llevado a un inusitado grado de dificultad y adicción.

No dudamos que SPELL BOUND y KNIGHT TYME están llamados a convertirse en dos nuevos éxitos que añadir a la abultada colección de méritos de su autor.

Ambos juegos funcionan a través del práctico sistema de iconos, mediante los cuales eliges tus opciones, alcanzas objetos, los examinas, o, si quieres los rechazas.

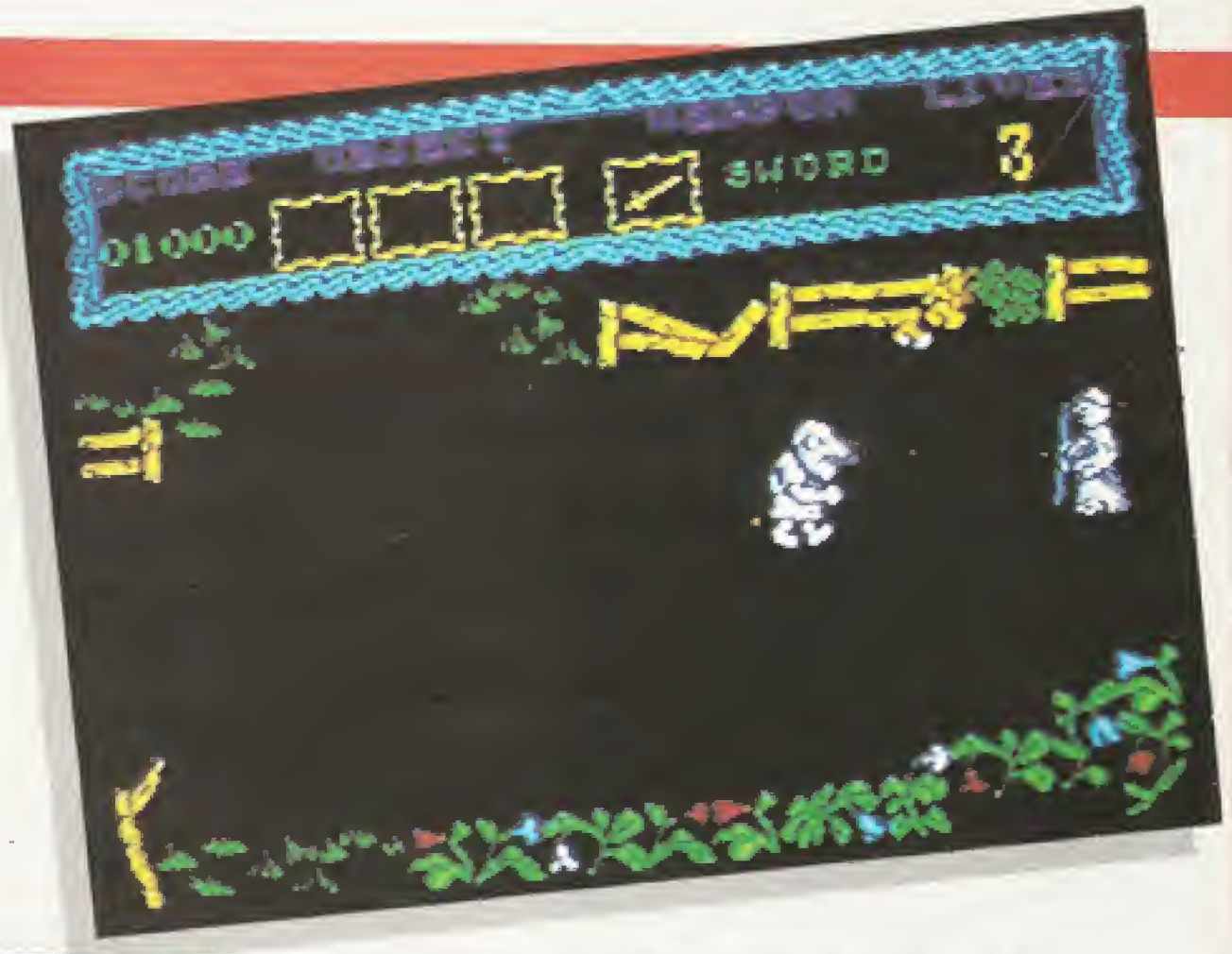
A pesar de ello reúne las características clásicas de una videoaventura. Estas últimas a pesar de la austeridad de sus trazos mantienen una calidad suficiente para que el juego se desarrolle en un cuadro de entretenimiento asegurado.

En este caso, tal como en el cine sucedió con *La guerra de las galaxias* y *El padrino* y, en literatura, con *El ingenioso hidalgo don Quijote de la Mancha* y *Los tres mosqueteros*, entre otros ilustres ejemplos, no sólo segundas partes, sino también terceras y más, siempre fueron buenas. Tenemos la seguridad de que estarás de acuerdo con nosotros.

ANIMACION	7
INTERES	9
GRAFICOS	8
COLOR	7
SONIDO	6
TOTAL	37

THE CURSE OF SHERWOOD

● MASTERTRONIC ■ AVENTURA MEDIEVAL

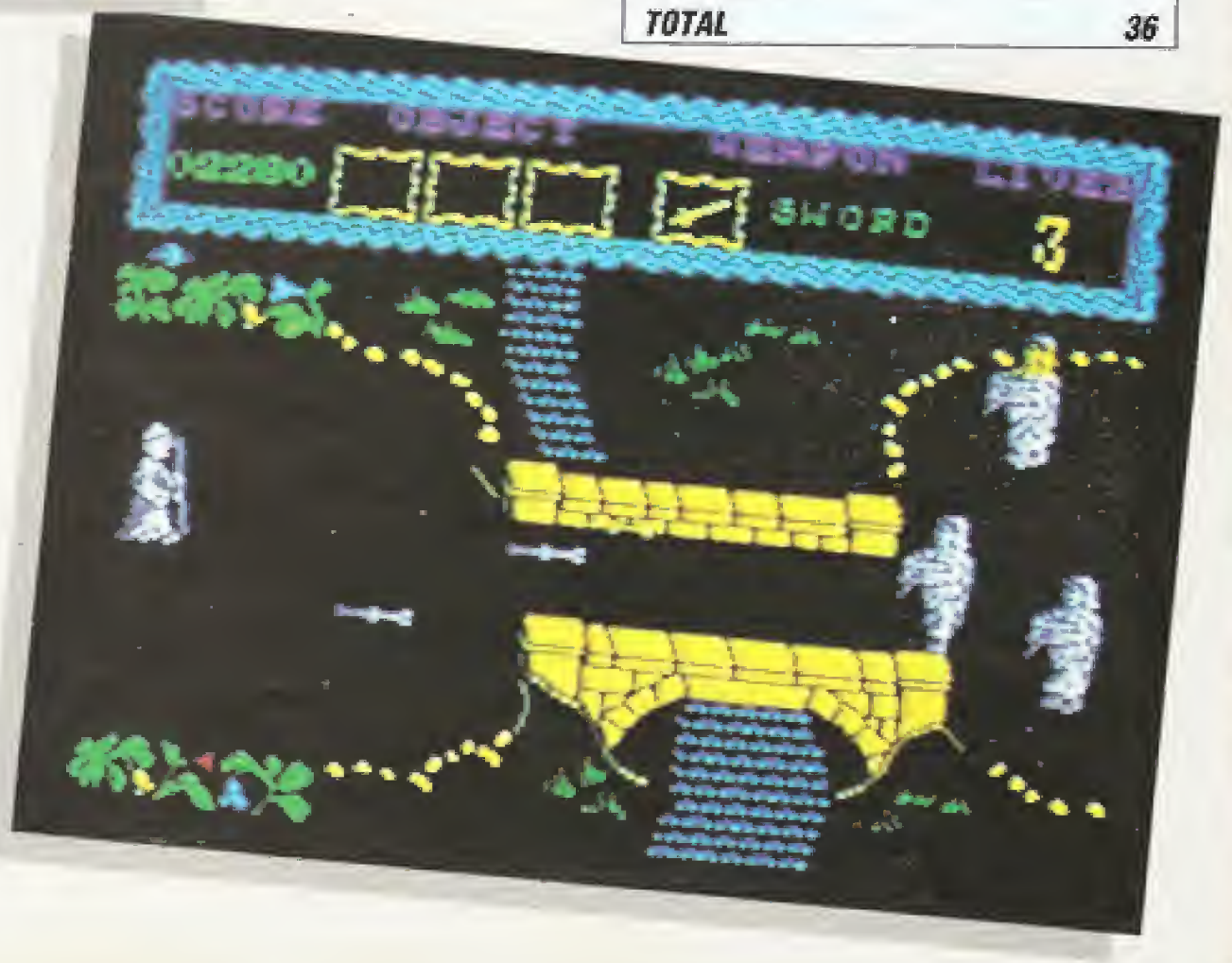


Un programa MASTERTRONIC de 500 pts., que bien vale la pena. Tanto su original diversidad gráfica, como la calidad del argumento, están más en la línea de un «número uno» que en la de una serie comercial de bajo precio. Su contenido puede resumirse así:
Los bosques de Sherwood han sido invadidos por fuerzas infernales que mantienen aterrorizada y sometida a la población. Robin y sus hombres

tratan de enfrentarse a ellas, como antes lo hicieran contra el malvado Juan sin Tierra, pero poco es lo que pueden hacer contra las fuerzas sobrenaturales, y como era de esperar, fracasan. La única esperanza del bosque reside en un pacífico monje, que debe encontrar el exorcismo adecuado para conjurar el mal, enfrentándose en su camino a los terribles ejércitos diabólicos.
Esqueletos piratas, espectros de soldados, arqueros, cuervos asesinos, y un sinfín de malvadas criaturas, harán todo lo posible para que nuestro protagonista fracase en su empeño.
Existen diversos tipos de armas, con diferentes grados de efectividad, y un elevado número de objetos repartidos a lo largo del juego, cada uno de ellos con una utilidad diferente. La clave del juego reside precisamente en saber usar el objeto adecuado en el momento preciso, y en llevar siempre el arma más eficaz contra cada enemigo.

En resumen, esta «maldición de Sherwood» —como se traduciría en castellano— gustará, sin ningún género de dudas, a todo el mundo, sobre todo a quienes disfrutaron en el cine y en la literatura con obras como *Ivanhoe* y *Robin de los Bosques*.

ANIMACION	6
INTERES	9
GRAFICOS	8
COLOR	7
SONIDO	6
TOTAL	36



10th FRAME

● US GOLD ■ SIMULADOR BOLOS

Parece que los deportes de masas van cediendo poco a poco su hegemonía frente al empuje de otras modalidades deportivas hasta ahora menos populares. En los últimos meses esta circunstancia se ha visto reflejada también en el software, con la publicación de títulos como *180* (simulador de dardos), PING-PONG, y BUMP-SET-SPIKE, a los que ha venido a sumarse 10th FRAME, un original simulador de bolos.

10th FRAME reproduce fielmente las distintas técnicas de lanzamiento de la bola, incorporando la posibilidad de calcular el «gancho» y la velocidad necesarios para derribar el mayor número posible de bolos.

Se puede elegir entre tres niveles de dificultad (infantil, amateur y profesional), y competir en partidas sueltas, o en campeonatos por equipos.

Aunque desde el punto de vista gráfico, el juego no es especialmente brillante (la verdad es que no han tenido muchas oportunidades de lucimiento sus autores), como simulador de bolos cumple sobradamente con su cometido, sobre todo si tenemos en cuenta las enormes posibilidades que ofrece para jugar torneos en que tome parte un elevado número de participantes.

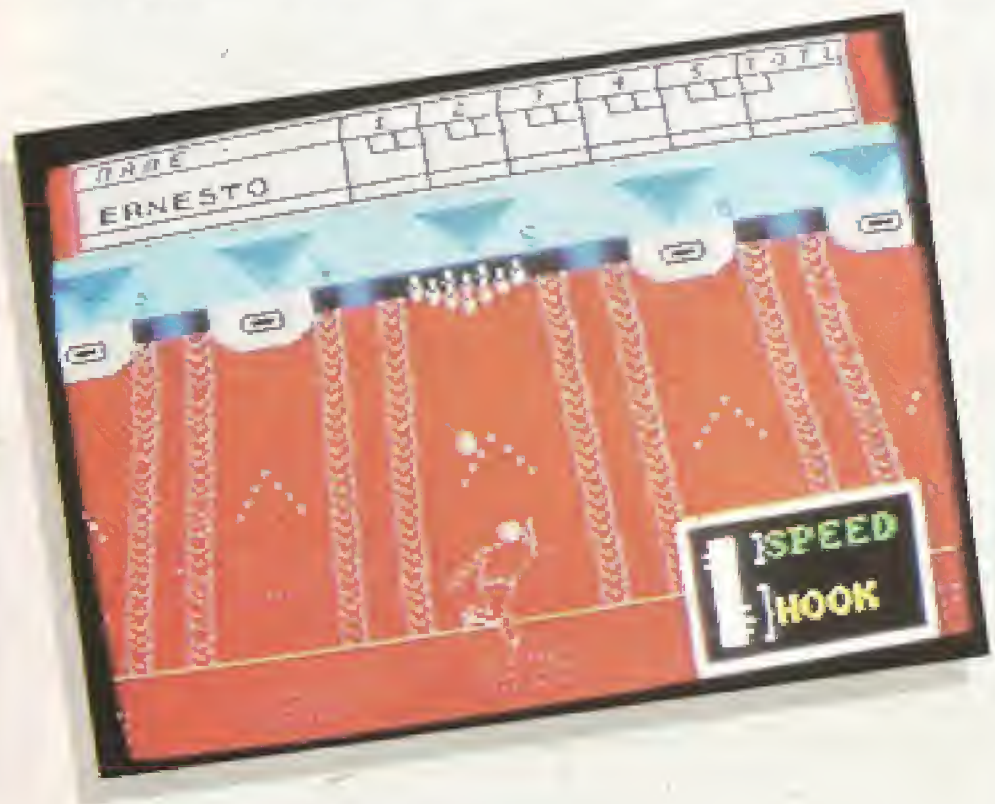
En definitiva, 10th FRAME es una excelente oportunidad para los aficionados a este deporte que no tengan la suerte de disponer de una bolera en casa.

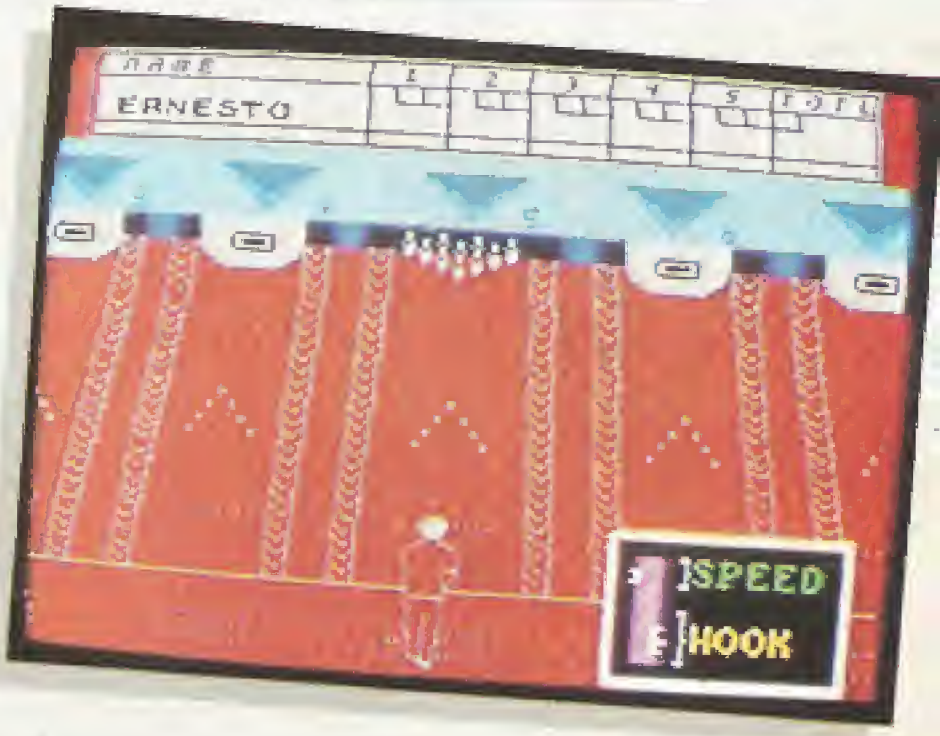
Como dato ilustrativo, te recordamos que en este deporte —el juego de cancha cerrada que más aficionados cuenta en Estados Unidos— la misión del jugador es lanzar una bola hacia un grupo de diez bolos dispuestos en triángulo, con el objeto de derribar el mayor número posible de ellos.

Los equipos suelen contar de 5 jugadores, cada uno de los cuales hace 10 tiradas, a dos bolas cada una. Gana quien haya logrado un mayor tanteo al final de los 10 turnos.

¡Ánimo, pues, amigos! Practicad con ahínco y perseverancia con este programa de simulación de bolos, y así estaréis en condiciones de desafiar con éxito a vuestros amigos.

ANIMACION	6
INTERES	7
GRAFICOS	6
COLOR	6
SONIDO	5
TOTAL	30





1942

● ELITE ■ COMBATE AERONAVAL

1942 es una interesante aventura mitad simulador, mitad arcade, ambientada en los combates aeronavales que tuvieron lugar en aguas del Pacífico durante la Segunda Guerra Mundial, entre las fuerzas de los Estados Unidos y los ejércitos aéreo y naval de Japón. Este programa de combate aeronaval de Elite es, en realidad, una adaptación del clásico juego de las máquinas recreativas del mismo nombre, en la que podrás ensayar tu habilidad como piloto de caza desde la cubierta de un portaaviones. Es interesante que sepas que durante la acción hay varios tipos de objetivos y un gran número de niveles diferentes, adecuados a tus gustos y a tus aptitudes en la manipulación del joystick. Aunque no puede decirse que sea tan fiel a la realidad como hubiéramos deseado, al menos hay que reconocer su mérito como arcade, y su elevado grado de adicción.

Veremos si con tu contribución los aliados vuelven a ganar, o si, por el contrario, esta vez vence Japón.

ANIMACION	7
INTERES	8
GRAFICOS	6
COLOR	6
SONIDO	6
TOTAL	33

COSA NOSTRA

● OPERA SOFT ■ ARCADE URBANO

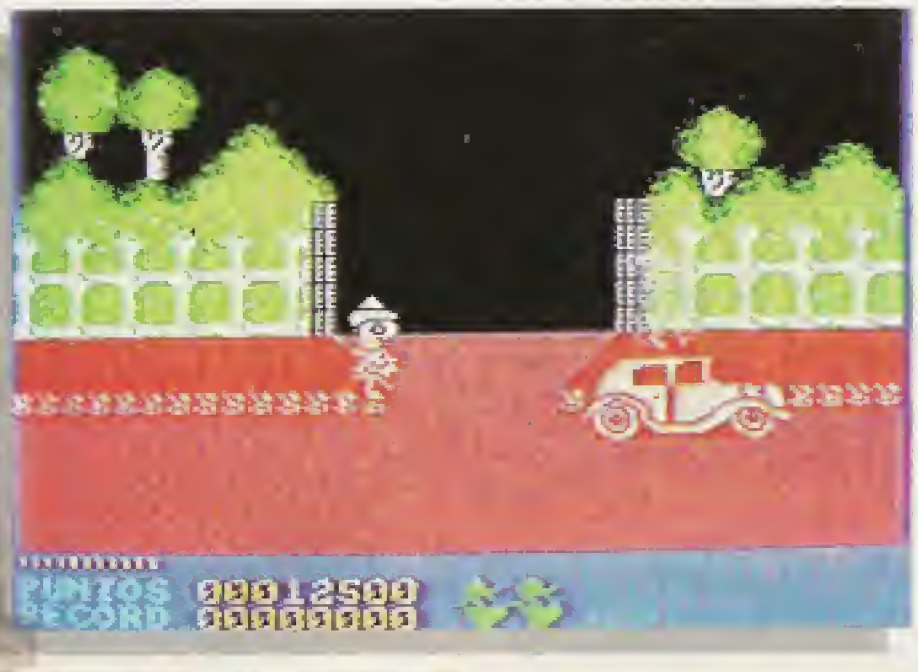
Cuando el Comisario Jefe de la policía puso a todos sus agentes al servicio de las bandas de malhechores, la mafia se apoderó completamente de la ciudad de Chicago. En poco tiempo, «El Padrino» y sus secuaces hicieron proliferar los garitos, las destilerías clandestinas, y los almacenes de mercancía robada donde planeaban sus actividades delictivas. Mientras los encargados de «velar por la ley y el orden» se repartían los beneficios de su participación en el negocio, los gángsters extorsionaban a los comerciantes, asaltaban los bancos y despojaban de todas sus pertenencias a los pocos que se atrevían a salir a la calle.
Para acabar con esta difícil situación, los representantes del gobierno que aún no habían sido sobornados decidieron contratar los servicios del super-detective Mike Bronco, un especialista en casos de divorcio que nunca había matado una mosca.

LA MISION

El objetivo del juego consiste en buscar en las 92 calles y cruces de Chicago a los cinco grandes Capos del hampa, y eliminarlos sin más contemplaciones. Ellos son nada menos que Ruddy Bulldog, Johnny Fandango, Tony Spaguetty, Franki Frondasio, y El Padrino, los más temibles gángsters de la historia. Naturalmente, para llegar hasta ellos primero será preciso enfrentarse a sus esbirros, que tratarán de acabar contigo por todos los medios a su alcance. Dinamiteros, policías corruptos, lanzadores de hachas, y la más variada fauna de hampones de los barrios bajos, aguardan acechantes en las ventanas y las esquinas el momento propicio para quitarte una de tus numerosas vidas.
Para defenderte de ellos y acabar con sus jefes, tu única arma es una pequeña pistola, bastante escasa de munición. Afortunadamente, cada vez que elimines a un policía aparecerá en su lugar un paquete de balas, que te permitirá mantener a punto el cargador cuando más lo necesites.

VALORACION

Tanto el argumento como el desarrollo de la acción, se caracterizan por una tremenda originalidad y un alto grado de interés, que convierten a esta vídeo-aventura en un arcade frenéticamente adictivo.
Desgraciadamente para los usuarios de Spectrum, la versión que se ha hecho para este ordenador ha perdido mucho en cuanto a su calidad gráfica. Las figuras de los personajes han sido notablemente simplificadas, y el colorido original se ha visto reducido también a su mínima expresión.
Aunque COSA NOSTRA no deja de ser por esta razón uno de los mejores programas de la temporada, no está de más hacer la advertencia, aunque únicamente sea para que los autores se esmeren un poco más en la próxima ocasión.



SUGERENCIAS

Procura economizar al máximo tus municiones. No olvides que si te quedas sin balas, no podrás matar más policías para obtener sus cargadores, y por tanto no te quedará más remedio que dar el juego por terminado.
Cuando en alguna pantalla aparezca un lanzador de hachas, retírate inmediatamente de su trayectoria para salvarte. Estos personajes siempre lanzan al frente su arma sin apuntar, por lo que bastará con no estar delante de ellos cuando lo hagan. Sin embargo, los dinamiteros de las ventanas y los policías de las aceras sí que apuntan hacia ti donde te encuentren, y por añadidura suelen ser bastante certeros. Procura disparar tú antes.
En algunos enclaves importantes, como la comisaría, o el ayuntamiento, tus enemigos saldrán por todas partes continuamente. Cada vez que mates a uno, aparecerá otro en su lugar. Por ello, debes pasar lo más rápidamente posible por estas pantallas.
Los barrios bajos de la ciudad son muy peligrosos, pero en ellos es más fácil alcanzar el objetivo del juego.

ANIMACION	*9*
INTERES	*9*
GRAFICOS	*7*
COLOR	*6*
SONIDO	*7*
TOTAL	*38*

TERRA CRESTA

● IMAGINE ■ ARCADE

Como muchos recordaréis, el año pasado publicamos el comentario de un magnífico arcade que nos dejó «pasmados» por su sorprendente calidad gráfica y su enorme grado de adicción. Se trataba de URIDIUM, uno de los títulos que mayor impacto causaron en el PC SHOW de septiembre. Pues bien, han tenido que transcurrir ocho largos meses de espera para ver publicado otro sensacional programa que, por fin, ha conseguido igualar la calidad del mencionado juego, o incluso superarle en algunos aspectos. Estamos hablando de TERRA CRESTA, un arcade que puede convertirse en uno de los mayores éxitos de IMAGINE.

El objeto del juego consiste en patrullar con una sofisticada nave la superficie de un planeta invadido por fuerzas alienígenas.

Los dos aspectos en los que más destaca TERRA CRESTA son los gráficos y el sonido, mucho más cuidados que en la mayoría de los arcades que se realizan por

conversión. En cuanto a la originalidad, debemos decir que muy difícilmente puede ser original un arcade «puro», como éste, pero a pesar de todo, tanto el diseño gráfico como el argumento contribuyen a paliar este pequeño defecto común a casi todos los «arcade».

ANIMACION	7
INTERES	9
GRAFICOS	9
COLOR	7
SONIDO	9
TOTAL	41

SUPER SOCCER

• IMAGINE ■ SIMULADOR FUTBOL

La firma IMAGINE ha pretendido decir con este programa «la última palabra» en materia de simuladores de fútbol, y, hay que reconocerlo, casi lo consigue.

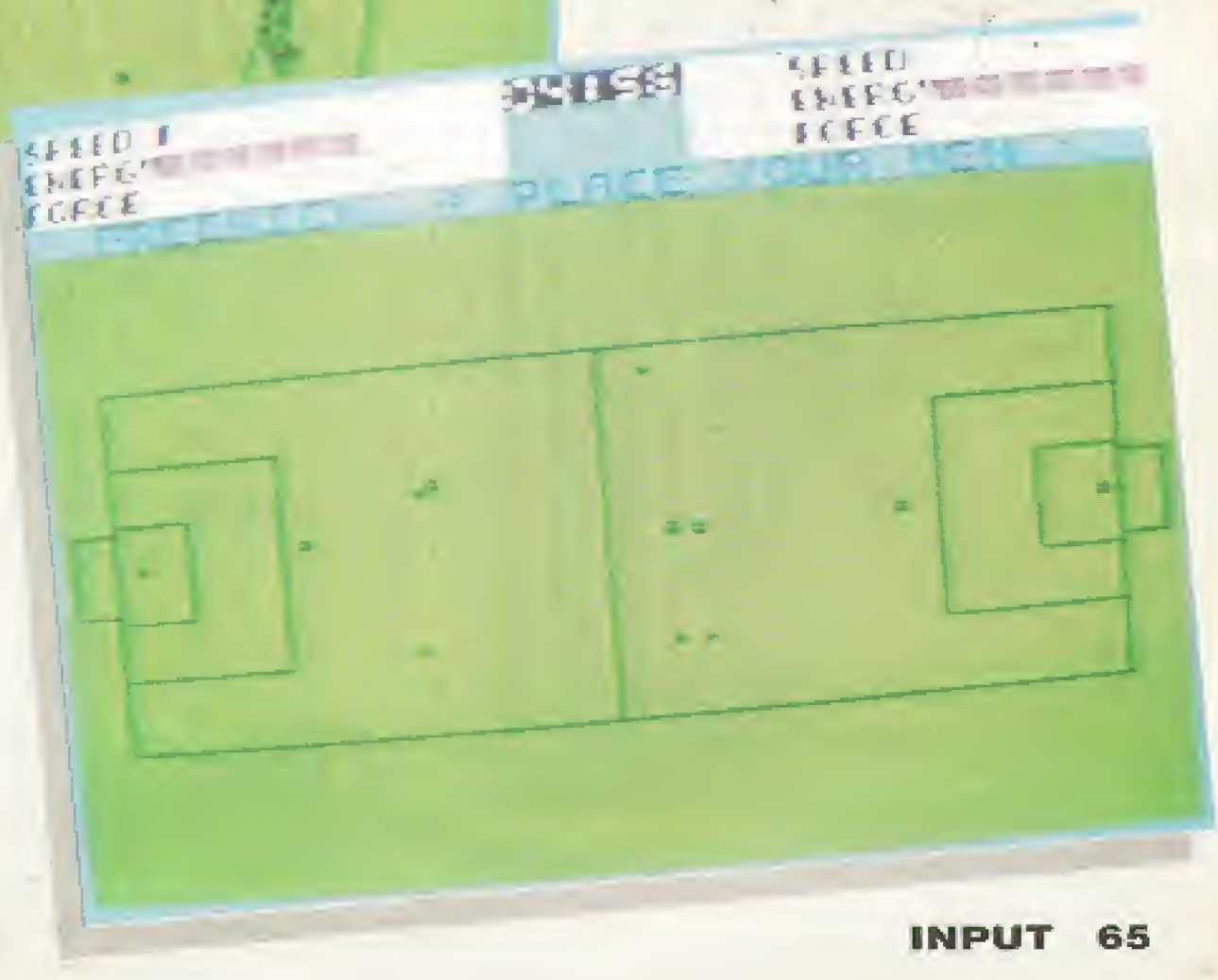
Decimos «casi» porque, si bien las innovaciones incorporadas son numerosas y muy acertadas, el resultado final no ha sido tan positivo como todos esperábamos.

La lentitud en los movimientos y la superposición incorrecta de los sprites, son feos detalles que ensombrecen otras lucidas características como los lanzamientos de banda, los penalties, las faltas, las entradas «en plancha», y los remates de cabeza.

No obstante, el juego en su conjunto tiene todo lo suficiente como para merecer una buena calificación global, y suponemos que apasionará y ganará un gran número de adeptos entre los aficionados a este espectacular deporte.

De más está que te sugiramos que organices un campeonato de liga en compañía de tus amigos, con o sin inclusión de *play off*, con este simulador del popular deporte lanzado por IMAGINE.

ANIMACION	8
INTERES	7
GRAFICOS	6
COLOR	6
SONIDO	6
TOTAL	33



EL ZOCO

Intercambio juegos también programas para el Spectrum 48K. César Hernández Cobo. C/ Gregorio Marañón, 5, 2.° C. Algeciras. Cádiz.

Vendo juegos de Spectrum a 700 pts, últimas novedades; Cobra, Infiltrator, The Great escape... etc. También los cambio. Vendo Copión OMNICOPI, 2. por sólo 1.000 pts. Francisco Cornejo Carretero. C/ Alfonso XII, 96. Badalona. Barcelona. TI: (93) 398 17 17. De 8 a 9 de la noche.

Vendo Spectrum 128K español completo, manuales, cables, teclado numérico independiente. 1 año y en perfecto estado. Interface de joystick programable y más de 35 juegos todos comerciales. Todo por 29.000 pts. Richard Gutiérrez. C/ Llago, s/n. Soto del Barco. Asturias.

Compro copión para ZX-Spectrum 48K, negociaré precio. También deseo contactar con chicos/as de Madrid para prestarnos juegos. Guillermo. TI: (91) 408 84 59. Madrid.

Vendo juegos para el Spectrum como PITFALL II (100 pts) todos de máxima calidad. Miguel Herrera Guzmán. C/ Formentera bl. D. 3.° 4.ª. Barcelona. 08016. TI: (93) 354 41 24.

Vendo Interface 2 tipo Kempston por tan sólo 2.500 pts. Javier González Álvarez. C/ Puerto Pajares, 5, 6.° D, Gijón. Asturias (33207). TI: (985) 39 83 28.

Vendo ordenador ZX-Spectrum plus, con transformador, grabadora, salida para monitor, TV y todos los cables de conexión. (Precio negociable). Juan Miguel, C/ Dña. Perfecta, 71. G. Canaria-Schammaan. TI: (928) 25 12 23.

Vendo 70 revistas de MICRO HOBBY de los números 23 al 93 por 4.500 ptas. TI: (91) 747 73 76. Carlos.

Interesado en contactar con usuarios del Spectrum para intercambio de Software. Josep M.ª Busquets. C/ Obispo Martín Ruano, 9. 25006 Lérida. TI: (973) 27 12 38.

Vendo Spectrum Plus, cassette, Interface tipo Kempston. Todo por 27.000 pts. Además regalo más de 50 juegos y 50 revistas. Fernando Sánchez. TI: 653 23 31. Madrid.

Vendo programas para ZX Spectrum. Novedades (Fairlight II, Avenger, Cobra, Uridium, Fist II, etc.). Precios muy interesantes. Vendo copión Turbo con instrucciones. Rafael Alcaide Jiménez. C/ Constitución, 20. San Feliu de Llobregat, Barcelona. TI: (93) 66 60 02 de 14:30 a 15 h.

Vendo Spectrum Plus prácticamente nuevo. Transformador, cables de conexión, joystick Quick Shot II, Interface Kempston 2 salidas, manual en inglés y castellano, la revista INPUT SINCLAIR del n.° 0 al 15, otras revistas y más de 200 juegos, la mayoría recientes. Todo por 45.000 pts. José Parejo. TI: (955) 25 97 47. Huelva.

Vendo ordenador ZX Spectrum Investronic con más de 200 juegos, cassette y todos los cables necesarios por 32.000 pts. o bien cambio por ordenador MSX 64K con todos los cables necesarios. José María. Manresa. Barcelona. TI: (93) 873 36 23.

¡LA MAS GRANDE AVENTURA ESPACIAL DE TODOS LOS TIEMPOS!

STAR WARS

LA GUERRA DE LAS GALAXIAS

CADA MES EN TU QUIOSCO

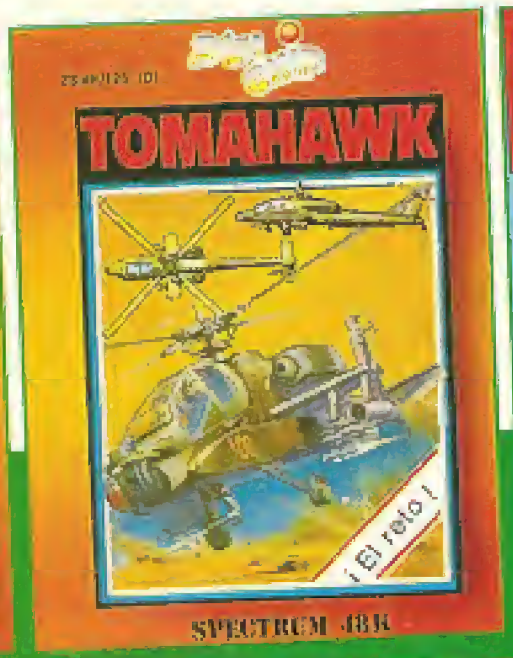
¡DOS COMIC-BOOKS USA AL PRECIO DE UNO!

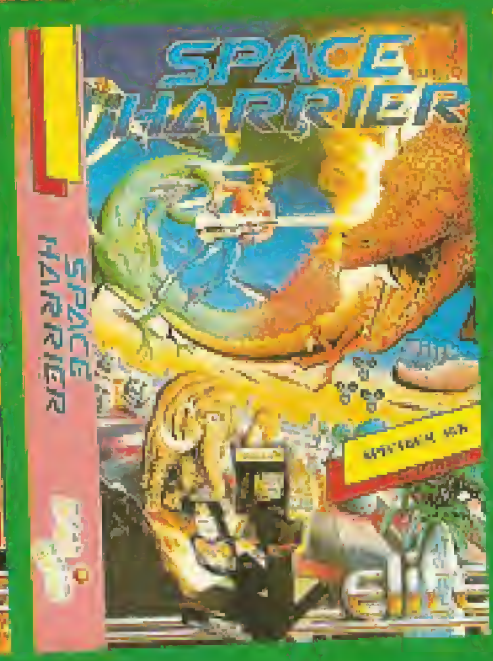
¡EL JUEGO MORTAL DE DARTH VADER!

Nº 7
140 PTAS.

COMICS forum

POCO RUIDO, MUCHAS NUECE





1.200 Ptas. (VERSION CASSETTE)

1.750 Ptas. (VERSION CASSETTE)

ZAFIRO SOFTWARE DIVISION Paseo de la Castellana, 141 28046 Madrid
Tel. 459 30 04 Tel. Barna 209 33 65. Telex: 22690 ZAFIR E

ZAFI CHIP

POCO RUIDO, MUCHAS NUECES